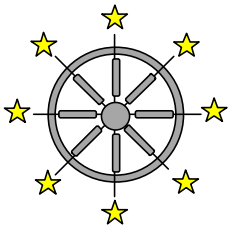
Kockums Computer Systems

S E A S P R I T E



Software Architectures for Ship Product Data Integration and Exchange

EP 22.390

Document Name: **Ship Product Model**

Document Date:	**28.07.98**
Document Owner	**KCS**
Document Author/s:	**Matthias Grau**
Task / Deliverable Number:	**T1.6 / D161**
Version:	**1.0**
Status	**Final**
Document Reference Number:	**SEASPRITE.KCS.ALL.WP1.22.03.98**
Circulation:	**SEASPRITE Consortium, EC Officers**

Contributions: DNV (Jochen Haenisch), LR (Tim Turner, Frank Stolte)

Document history

Version	Date	Changes
0.1	12/14/98	1. This document has been established based on N 498 'AP Development Guidelines for Shipbuilding' (also known as the 'Cookbook') dated October 19, 1997. 2. Editorial and technical changes incorporated to reflect the present status of the Ship Common Model. 3. Section about features added as provided by Jochen Haenisch, DNV.
0.2	19.06.98	1. Ship materials section completed 2. Product structure by space section added as provided by Tim 3. General characteristics section added as provided by Tim
0.3	01.07.98	1. new and improved EXPRESS-G pictures 2. Connectivity section removed 3. Library element references sub-section added
1.0	28.07.98	1. Location concepts section updated as provided by Frank 2. Moulded forms section updated as provided by Frank 3. Ships section updated as provided by Frank 4. Configuration management section added as provided by Jochen 5. Building Block section updated
	04.12.98	SeaSprite copyright statements removed to allow for ISO distribution (Jochen Haenisch).

Table of Contents

Table of Figures

1 Executive summary

This document describes the approach to integrate the different ship data models that are currently developed as ISO 10303 (STEP) application protocols into one Ship Product Model. One of the most important preconditions for integration is to provide a common basis. This common basis for the ship data models is referred to as the Ship Common Model. Consisting of different parts it serves as a generic base model to be specialised in the different data models on one side while providing common data structures ready to be applied in the data models on the other side. The approach addresses the overlaps that exist between the ship data models and avoids redundancies while aiming for consistency and interoperability.

A short introduction into the field is followed by a detailed description of the major Ship Common Model parts, such as Modelling Framework, Domain Models and Common Utilities. The integration of data structures implies the need to make a meaningful subdivision of the overall model into manageable parts. A presentation of currently existing/discussed approaches can be found at the end of the document.

2 Introduction

2.1 THE NEED FOR A SHIP PRODUCT MODEL

A product model is a set of objects and relationships between the objects. While the objects describe the assemblies and components of the products, the relationships describe the architecture of the product. As the base for different activities, from idea to final product, the product model is the key to successful product realisation. As a knowledge base it contains geometric and technical data but can also refer to company specific information, product background, history, synthesis and analysis results, reasons for decisions etc.

The ship product differs from many other products by

- it’s complexity - a variety of different production branches and organisations have to collaborate in a co-ordinated way
- it’s uniqueness - ships are usually built in very small series of 1 to a few

which positions it closer to a power plant that to a car while being a vessel. This creates specific requirements on the handling of the product data.

The complexity of the ship causes the fact that not all information about this product is possible to be handled by one specific software but a variety of systems in the different organisations involved and even within one organisation (e.g. the ship yard) is necessary to create, process and maintain the data. This creates the problem that this data needs to be shared or even exchanged between the systems which either requires a lot of time consuming and error prone conversions or an underlying data structure that is commonly available to the different systems.

The complexity also creates the need for co-ordination between the numerous different organisations and between the departments within these organisations which requires configuration management information to be available and maintained by the systems.

Different versions of a design for instance can exist in parallel at the same time and are used by different organisations (e.g. the yard's design office and design subcontractors) due to concurrent engineering.

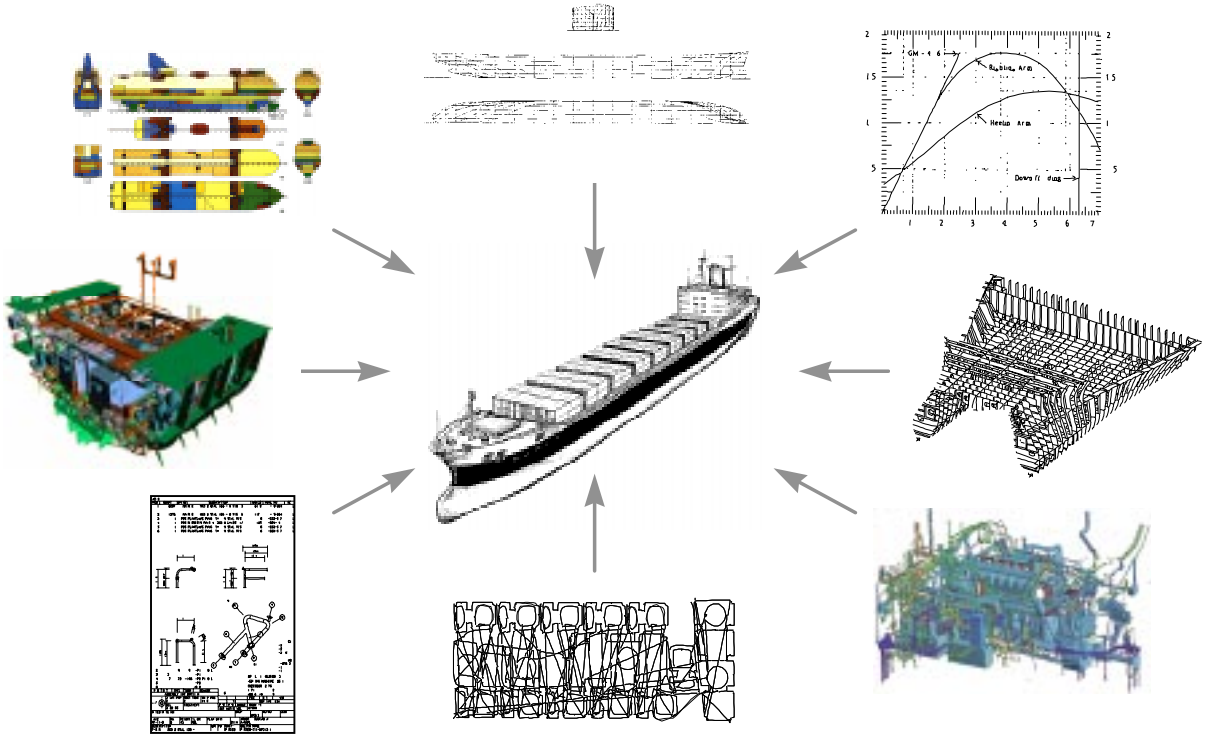


Figure 1: Ship Product Model

The small series that are usually built in the shipbuilding industry require this complex and interlinked information to be maintained in a very efficient and open way.

All these facts make it necessary to have a data structure that

- describes as complete as possible all information related to a ship
- in a way that is neutral, commonly agreed on and available and
- that allows for the complexity and concurrency that is usual within this industry.

The current approach to integrate the ISO 10303 shipbuilding application protocols is driven by the above mentioned requirements. This document shall contribute to that development in describing and further developing that approach of a Ship Product Model as part of Seasprite.

2.2 ONGOING SHIPBUILDING RELATED DEVELOPMENTS IN ISO 10303

ISO 10303 is an International Standard for the computer-interpretable representation and exchange of product data. The objective is to provide a neutral mechanism capable of describing product data throughout the life cycle of a product, independent from any particular system. The nature of this description makes it suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases and archiving.

This International Standard is organised as a series of parts, each published separately. The parts of ISO 10303 fall into one of the following series

- description methods
- integrated resources
- application interpreted constructs
- application protocols
- abstract test suites
- implementation methods
- and conformance testing

The series are described in ISO 10303-1.

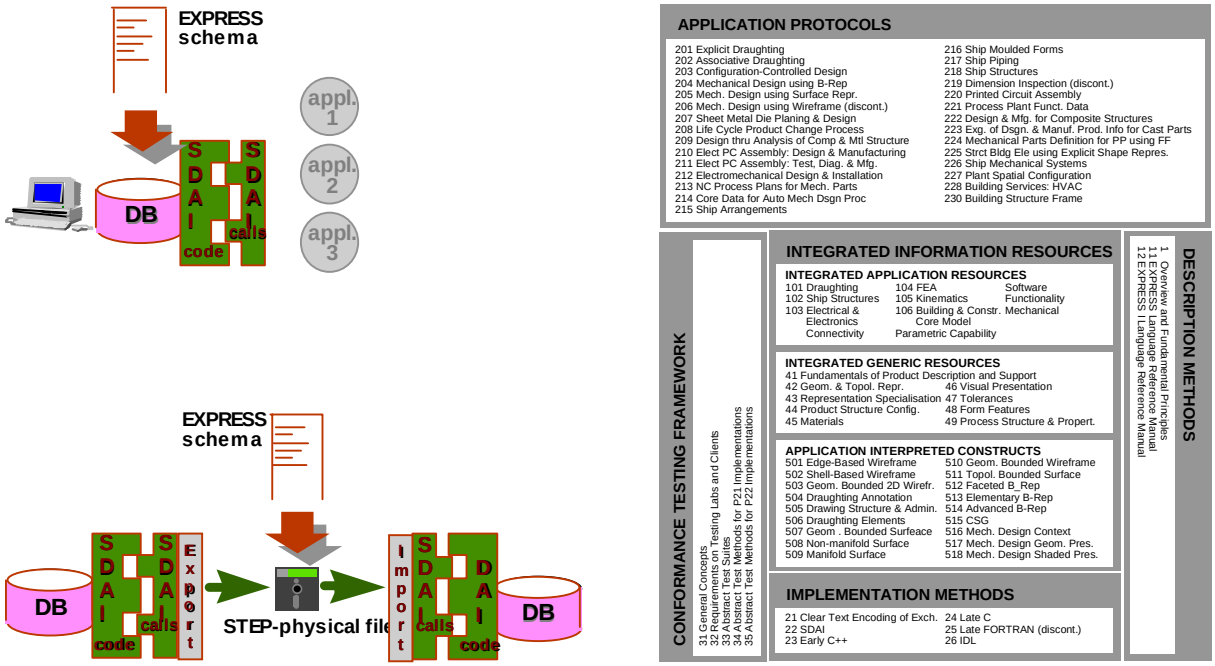


Figure 2: ISO 10303 - Principle Overview

The series of industrial application protocols assumes that the product model can be divided into separate parts that each cover a key element of the product for its whole life cycle. This also holds for the shipping and maritime area where the assumption is that the product model of a ship - the *Ship Product Model* - can be divided into separate parts for the same purpose. An overview about this subdivision gives Figure 3.

The reason for doing this is as much to do with distribution of modelling work as it is with the need to exchange subsets of the product model between agents in the marine industry, let alone the practical aspects of exchanging the data associated with an entire ship.

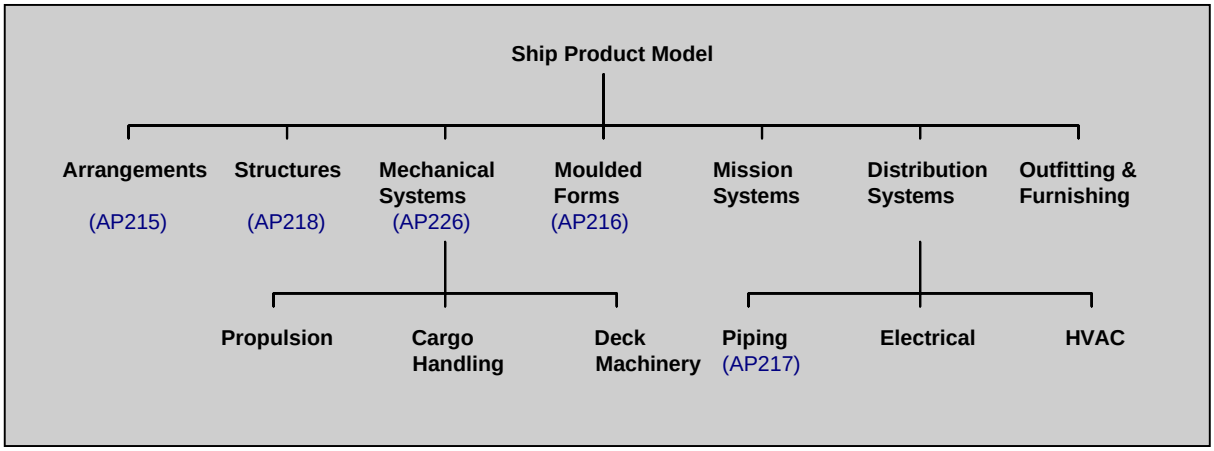


Figure 3: Ship AP Planning Model

These key elements are:

- ship arrangements
- ship structures
- ship mechanical systems
- ship moulded forms
- ship mission systems
- ship distribution systems
- ship outfit and furnishings

Each separate system is described by one or more different application protocols. The full series of possible shipping application protocols (AP) is shown in Figure 3 indicating those which are under development at the present time. See also 5.3 for an overview about the scope of the shipbuilding APs actually being developed.

Simply put, an AP has three major parts:

- an Application Activity Model (AAM) to describe and decompose the activities, input and output objects, controls and modifiers
- an Application Reference Model (ARM) to describe the information objects required, their structure and attributes
- an Application Interpreted Model (AIM) to map the requirements to the types of objects understandable to other CAD systems

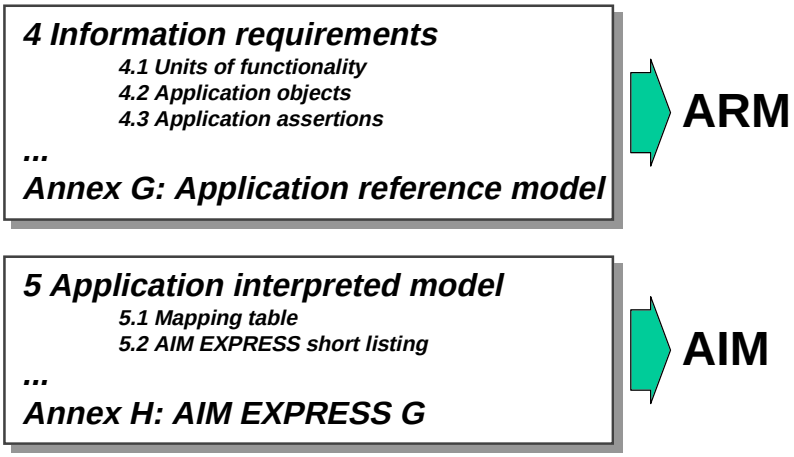


Figure 4: Structure of an AP

However, because applications will probably need to be able to handle information about different parts and areas of a ship - perhaps over a number of iterative exchanges over time - not restricted to just a single AP, the data (and therefore, the ARMs) representing the various parts of the ship must already be able to be integrated. This requires that there is an overall mechanism, around which all the shipbuilding AP's can be integrated. This mechanism should be central to each of the AP's structure such that the information is structured and organised in a consistent and similar manner and is known as the *Ship Common Model*.

There are several by-products of this. For example, the complexity of the models should reduce whilst the understandability of the different AP's among modellers increases, allowing for both easier interoperability and integration of the overall product model. Information can also be navigated and retrieved in a similar manner, regardless of which AP is being used, through the use of such a mechanism and can help the process of interpretation and development of the AIM.

3 The Ship Common Model

3.1 Introduction to the Ship Common Model

The *Ship Common Model (SCM)* forms the basis for all shipbuilding AP ARMs under development and to come. It provides a *Modelling Framework* for the APs that represent the *Ship Product Model*, a set of independent and re-usable product-structure *Domain Models* that are required for more than one Application Protocol, as well as a set of commonly used constructs refereed to as *Common Utilities* such as those used for configuration control and management concepts. The goal of the *Ship Common Model* is to ensure to the integration and overall consistency of the ARMs of the different ship APs. Thus, it is a means of integrating the requirements specified to a uniform conceptual model.

The ISO AP Development Guidelines [AP Guidelines] state that each AP should have a data planning model defined in terms of the major units of functionality used in the AP and should show the relationships between these and the framework. This provides an implicit requirement that each AP should conform to the framework and be consistent with the modelling techniques that have been used.

3.2 Overview of the Ship Common Model

As mentioned above, the *Ship Common Model* can be split into three parts, the *Modelling Framework*, a set of *Domain Models*, and a set of *Common Utilities* such as configuration management concepts and measurement units.

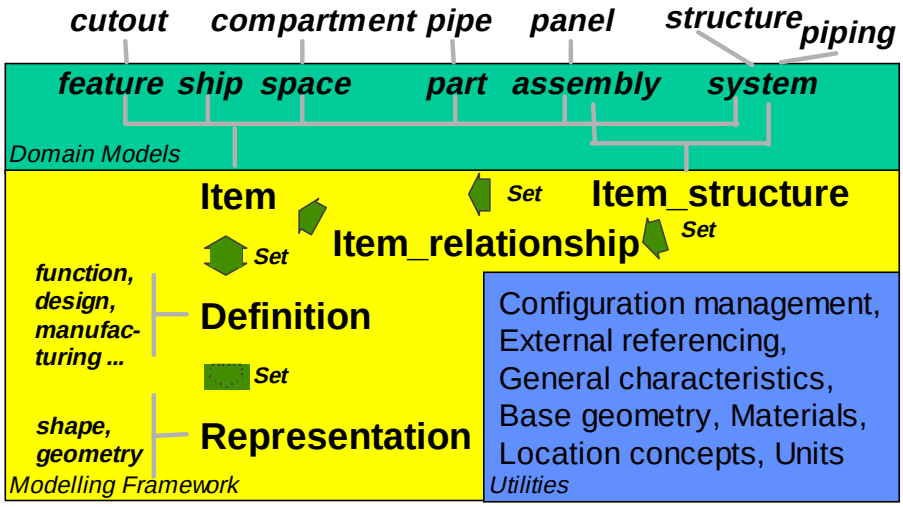


Figure 5: Parts of the SCM

3.2.1 Modelling Framework

The *Modelling Framework* as part of the *Ship Common Model* provides the realisation of the general idea of

- how to relate concepts
- how to define their properties
- how to represent them

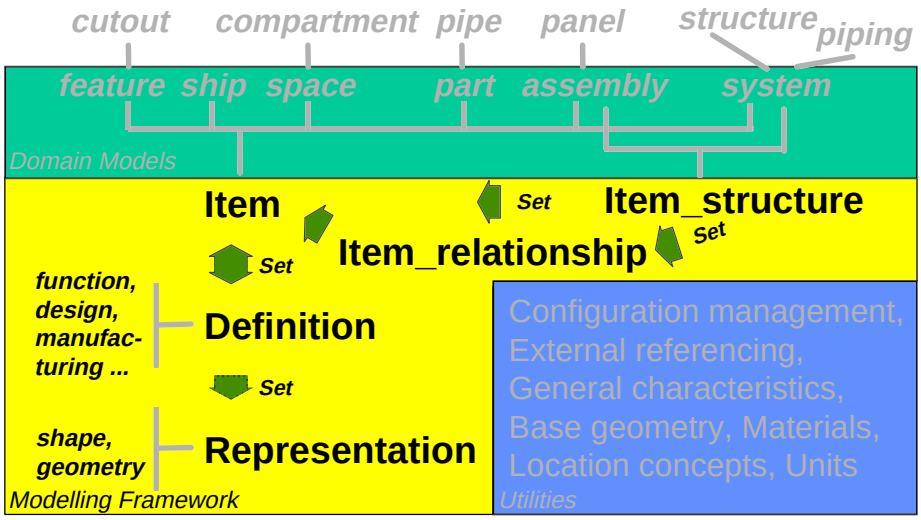


Figure 6: Modelling Framework

Effectively, this high level approach forces the product model to be split up across the main constructs of the *Modelling Framework*, namely

- items
- definitions
- representations

whilst being linked via generic relationships. One of the benefits of this approach is that it enables a better management of information such as need to organise the data according to different viewpoints and in the representation of life-cycle dependent requirements.

3.2.2 Domain Models

On top of the *Modelling Framework*, the *Domain Models* provide a set of templates for the organisation of the product being modelled along a number of different axes or views, such as

- Parts
- Features
- Product Structure by System
- Product Structure by Space
- Product Structure by Assembly

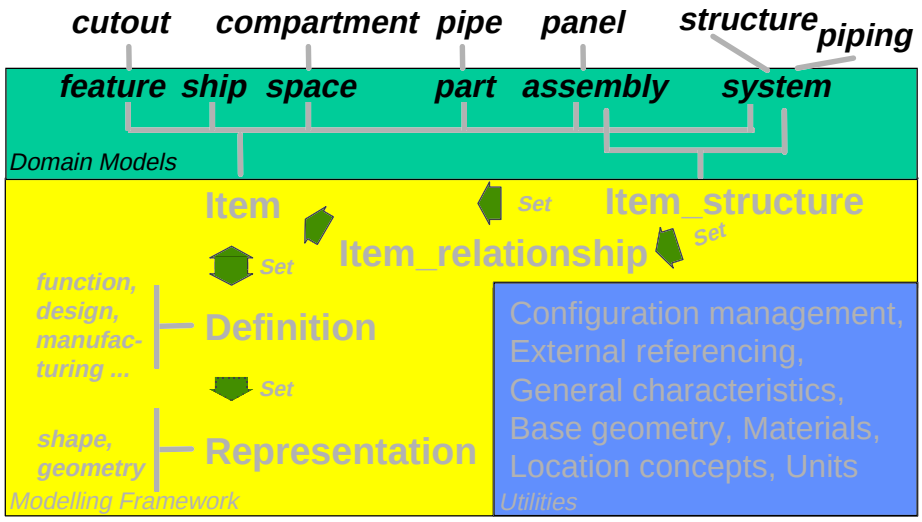


Figure 7: Domain Models

However, the templates also provide a set of implicit modelling techniques for the organisation of the product data. These *Domain Models* represent atomic concepts and generic structures which - whilst conforming to the framework - allow an AP to organise the data of the product (through a process of specialising the generic concepts) according to its needs. In doing this the need to invent disparate modelling strategies for each AP product structure is negated. The benefit of this approach is reduced modelling effort, consistency, conformity and interoperability with the other AP's that already conform to this approach.

The reasons for organising the data model along the lines of the templates are not focused entirely upon the need to reflect real world objects, but also from the needs of the framework to support interoperability, integration and consistency.

It is not intended that each of the *Domain Models* is used by an AP. The type of structuring needed for the *Product Model* should become evident through the initial modelling work, although it is not uncommon for a number of structuring techniques to be used in an AP, and the generic domain model specialised accordingly.

3.2.3 Common Utilities

The *Common Utilities* are a group of constructs that will be required by most AP's. They differ from the *Modelling Framework* and the *Domain Models* through the fact that for the majority of cases, the *Common Utilities* are ready for use and do not require any further specialisation for use in an ARM. Many have been created specifically for shipbuilding although some may be able to be used externally. The *Common Utilities* group together the following *Units of Functionality*:

- *Ship General Characteristics*; including the basic ship's length, breadth, type and class
- *Location Concepts*, including the global co-ordinate system, local co-ordinate systems, spacing grids etc.
- *Ship Moulded Geometry* such as ship point, curve and surface
- *Configuration Management*, like approval, versioning and change management

- item *Ship* itself since all data defining the product need to be related to the product which is ship
- *Measurements*, the types of measures to be used such as metres, kilogram or second etc.
- *External Referencing Mechanisms* allowing to point to information that is outside of the scope of a certain data exchange

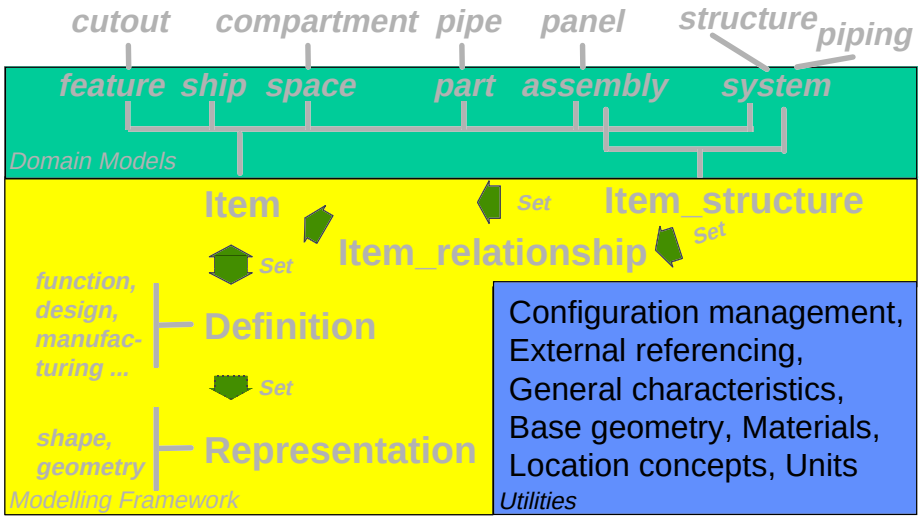


Figure 8: Utilities

3.3 MODELLING FRAMEWORK

The *Modelling Framework* is the "backbone" of the different ARMs. It can be reused in the APs and specialised by sub-typing from the concepts presented and described in this section.

3.3.1 Items and Definitions

There are two major tasks when creating a part of a product model:

- defining concepts by specifying their properties
- describing how concepts are related to each other.

The properties of a concept are, in terms of modelling in EXPRESS, attributes of an entity. Instantiating a concept would therefore require every non-optional ‘entity’-attribute to be available.

As the amount of information known about a concept - it’s properties - usually grows during it’s lifecycle and therefore is only ‘complete’ at the end of it’s life time this dependency is often not desired. It can be removed by separating the concept from it’s properties - just from the modelling point of view - and allowing the concept to exist in an incomplete state until all of it’s properties are specified (i.e. making it just a placeholder without attributes, but able to join relationships to other concepts).

Thus, in the *Ship Common Model* concepts may exist while the level of completeness of the real world objects or ideas that they represent is reflected by the definitions that carry the properties. The SCM provides for this through the use of two constructs

- *Item* - stands for the concept
- *Definition* - holds the properties

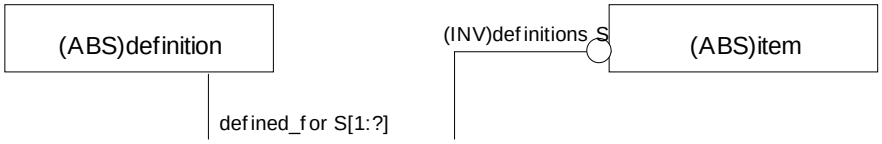


Figure 9: Relationship between Definition and Item

While a *Definition* must be *defined for* an *Item* (i.e. there is an existence constraint saying ‘no *Definition* without associated *Item*’) an *Item* may exist without any *Definitions*. This so called "defined_for" relationship shall be the only relationship between *Items* and *Definitions*, no other attributes shall cross this boundary.

An *Item* has an attribute called "id". This is a life-time identifier of the concept. In some contexts this identifier is referred to as tag-number, this is an identifier for a function rather than for an instance.

The *Ship Common Model* currently distinguishes among the following *Definitions*:

- <Design_definition>
- <Functional_definition>
- <General_characteristics>
- <Lightship_definition>
- <Loading_condition_definition>
- <Manufacturing_definition>
- <Parametric_definition>
- <Ship_material>
- <Structural_part_class_definition>
- <Technical_description>
- <Tonnage_definition>.

This collection shows that the level of *Definition* at least captures the following categories of properties of objects:

- characteristics
- functionality
- life-cycle.

3.3.2 Item Relationships and Item Structures

Also the relationship between two concepts can be modelled as an entity (such as a ‘relationship-entity’) with then two attributes of type *Item*.

This allows to create instances of concepts and to describe their relationships - *Item_relationship* - on every level of completeness.

Item_relationships may, for instance, be used to model that:

- one concept can be connected to another concept;
- one concept can be bounded by another concept;
- one concept can be derived from another concept.

Figure 10: Item_relationship

Note that relationships between concepts that only carry existence constraints without any additional information should not be modelled as *Item_relationships*, because existence constraints are implicitly provided by the EXPRESS modelling language. The following relationships may for instance better be modelled as attributes to *Item* than as an *Item_relationship*:

- one concept can be realised by another concept;
- one concept can be part of another concept;
- the existence of one concept can depend on the existence of another concept.

While *Item_relationships* describing relationships between concepts there is a need for a container-like construct able to collect concepts and/or their relationships from a specific point of view. Such a container is provided by *Item_structure* (see Figure 11: Item_structure) which functions in a similar way as *Item_relationship*. This allows to collected concepts (as well as their relationships) into *Item_structure* without having any property defined.

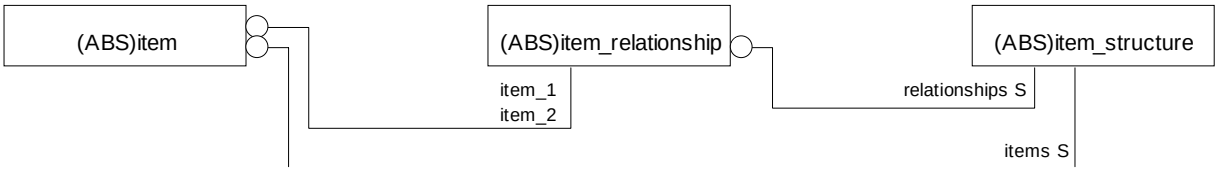


Figure 11: Item_structure

These three entities (Item, Item_relationship, Item_structure) are located on the same level of completeness, i.e. no property has to be defined for the concept represented by its placeholder *Item* to say it exists, it may be related to another concept, and it may be collected into structures within certain contexts.

In fact there is no reason for an *Item_relationship* not to have properties that are special to this relationship and that are not associated with the one or more concepts taking part in this

relationship. This makes it desirable to allow *Item_relationships* to have *Definitions* as well. And why shall an *Item_structure* not have properties special to this collection and not associated with the *Items* or *Item_relationships* it holds?

This requirement asks for a solution that allows *Definitions* to be defined for *Items*, *Item_relationships* or *Item_structures*. The solution is to move this common behaviour one level up and let the three entities inherit the ability to be defined is this way. This is provided for in the *Ship Common Model* by the *Definable_object* construct as shown in **Error! Reference source not found.**.

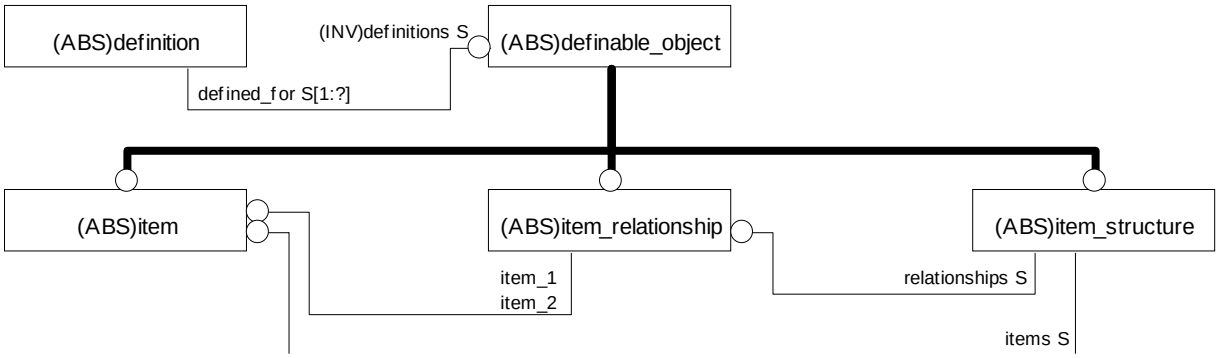


Figure 12: Relationship between Items, Item_structures, Item_relationships and Definitions

3.3.3 Representations

Every property of a concept, and therefore, every *Definition* of a *Definable_object*, may be described in several different ways.

The shape of a plate for example, may be described by parameters length, breadth and thickness, or by topological references to other plates this way specifying the boundary information and the thickness, or as explicit geometry - a curve describing the boundary contour - and the thickness. Each type of description may be useful in a different context or under certain conditions. All have in common that they describe the shape of a plate while doing this in a more or less implicit way. For viewing purposes e.g. it is desirable to have some explicit information available that ‘shows’ what the *Definition* is meant to be without being forced to process the description first.

Therefore it is worth to distinguish between the more or less implicit description of a concept and the explicit result of that description.

In general, the explicit result of the description of a concept shall be modelled as a *Representation*. A *Representation* shall carry the explicit result of the description of a concept.

A typical relationship between a *Definition* and a *Representation* is shown in the figure below.

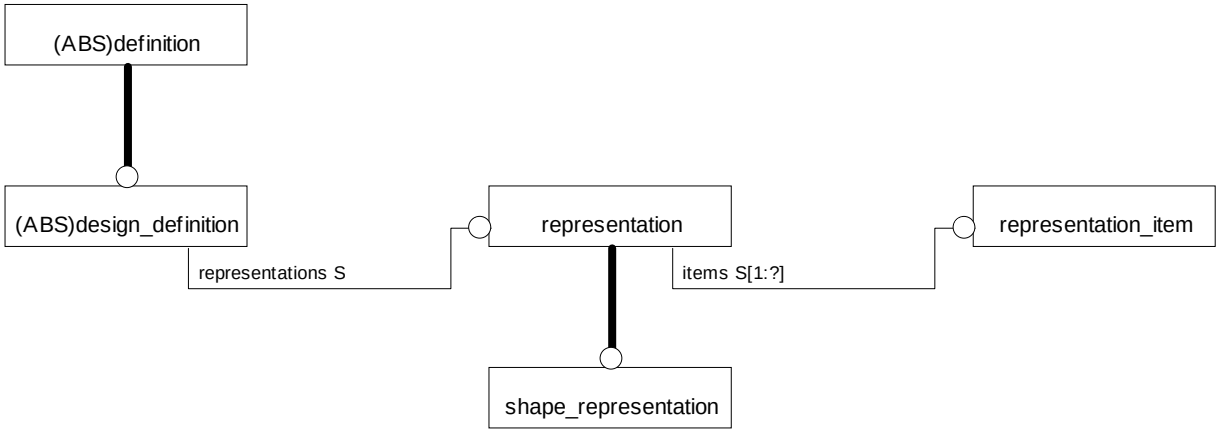


Figure 13: Relationship between Definitions & Representations

Other subtypes of **Definition** may be constrained to be certain subtypes of **Representation**.

The *Representation* concept itself is identical to the Representation in ISO 10303-43. Thus, the *Representation* in the *Ship Common Model* consists of a set of *Representation_items*. It is up to each *Representation* specialisation to constrain the valid types of *Representation_items*. *Representations* may also be related to each other with or without transformation by relationship entities.

So far the Ship Common Model uses only *Shape_representation* as specialised *Representation*. This however will be extended as the requirements for representations develop.

3.4 DOMAIN MODELS

3.4.1 Parts

A *Part* is an *Item* that is said to be made of a material. The reason for their existence is to be able to restrict special *Item_structure* subtypes (such as *System* or *Assembly*) to only collect *Parts* and not every *Item*. Without *Parts*, it would not be possible to make this restriction and by this e.g. an *Assembly* would be allowed to collect also every other kind of *Item* (such as *Moulded_form*, *Space* or *Ship*). The same is valid for the *Item_relationship* subtypes.

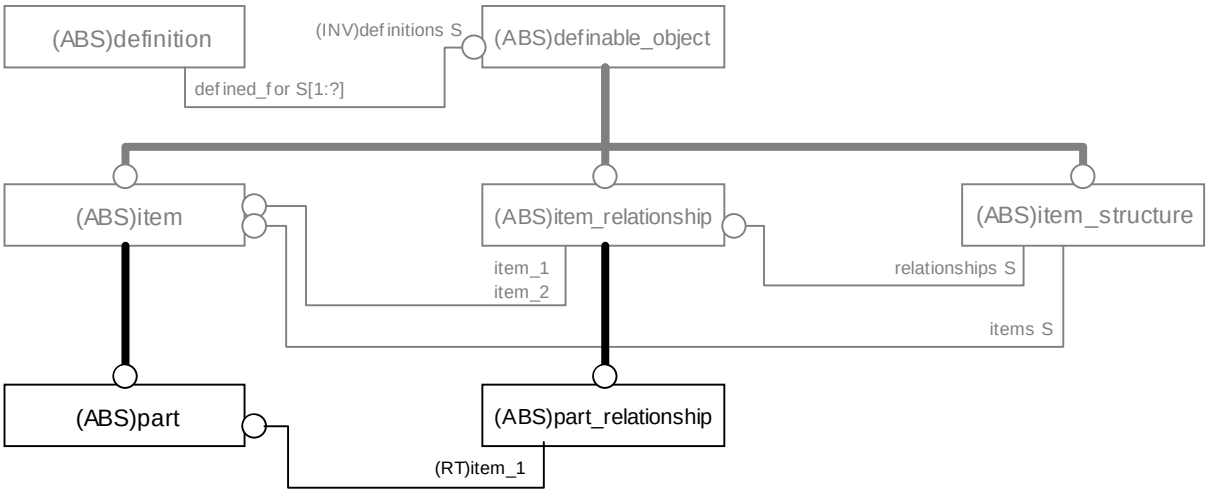


Figure 14: EXPRESS-G of Generic Product Structure - Part Link

This means *Part* is the atomic *Item* in all *Item_structures* collecting *Parts*. Typical *Parts* are

- Pipe_parts like Valve or Pipe
- Structural_parts like Plate or Profile
- Machinery_parts like Equipment or Mechanical_product

3.4.1.1 Part

Parts are the atomic pieces a product is made of. Therefore parts are supposed to be the leaf nodes of the hierarchical structures.

<Part> itself only carries the identity of a part while the properties that are specified during the different life cycle stages are attached via <Definition>s. Typical examples of parts are:

- <Pipe >,
- <Equipment >,
- <Plate>,
- <Profile>,
- <Machinery_part>

<Part>s may be related to each other or to other <Item>s using <Part_relationship>s. These could e.g. describe the connectivity, adjacency or symmetry of two <Part>s or could define that one <Part> is bounded by another <Part>.

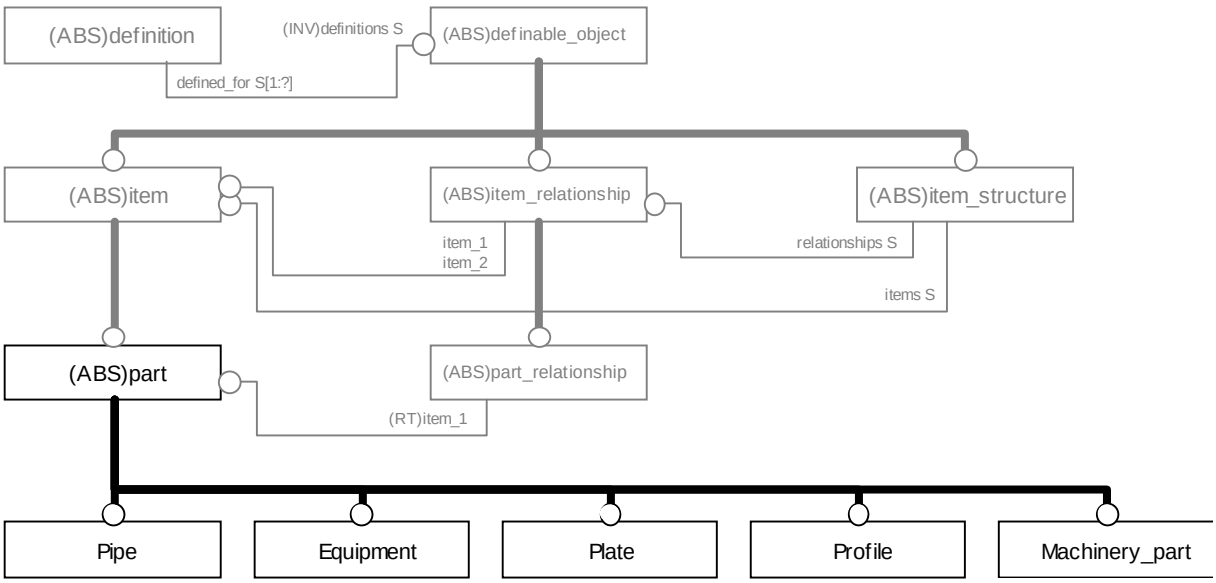


Figure 15: The parts model

3.4.1.2 Part Definitions

All lifecycle related properties a <Part> can be provided by relevant subtypes of <Definition>. The following definitions are currently available partly as abstract supertypes intended to be specialised by APs:

- <Functional_definition> (functional view)
- <Design_requirement> (design view)
- <Design_definition>, <Library_design_definition> (design view)
- <Ship_material> (design view)
- <Manufacturing_definition>, <Library_manufacturing_definition> (production view)

These definitions reference the one or several <Part>s that they are valid for by means of the <defined_for> attribute which they inherit from entity <Definition>.

3.4.2 Features

Something may be regarded as a feature if it fulfils the following requirements [Shah 90b]:

be a physical constituent of a part;

be mappable to a generic shape;

have engineering significance;

have predictable properties.

These characteristics identify what may be best called abstract feature [Shah 91]. In the ship product model this is the entity <Feature>. In literature a lot of different kinds of features have been proposed: functional features e.g. [Giacometti 90a,b], assembly features [Sodhi 91], mating features, and physical features [Kiriyama 91]. Such nature of a feature often evolves during the engineering process. At its current state the ship product model defines only one domain of features, i.e. <Structural_feature>, and covers only one life-cycle phase, i.e. design. Other feature definitions may be required in the future, such as for engineering analysis,

process planning, manufacturing, and inspection. When widening the scope of the ship features other STEP-work may need to be considered, such as ISO 10303-224, Mechanical product definition for process planning using machining features.

The feature concept as applied to the ship product model may become clearer by comparing a few statements:

Features can be viewed upon as information sets that refer to aspects of form or other attributes of a part, in such a way that these sets can be used in reasoning about design, performance and manufacture of the part or the assemblies they constitute [Salomons, 1995].

Other definitions confirm the central role of shape for features:

A generic shape which carries some engineering meaning [Wingerd 91].

However, for the general case the ship product model does not require a feature to have shape. It follows, thus, definitions like:

Recurring patterns of information related to a part description [Shah 90a].

A carrier of product information which may aid design or communication between design and manufacturing, or between other engineering tasks [Shah 90b].

3.4.2.1 Feature

<Feature>s are subtypes of <Item>. Like <Part>, <Feature> itself only represents the idea of a feature including a global identifier, a name, description, reference to documentation, and a pointer to the ship that it is valid for. Typical examples of features are:

- <Corner_cutout>,
- <Edge_cutout>,
- <Interior_cutout>,

which may represent profile penetrations and ratholes for the <Structural_part>s or <Structural_system>s of a ship.

Features may be related to each other or to another <Item> using <Feature_relationship>. <Feature_relationship> may for example be applied to indicate that two <Seam>s (which are features) are parallel or that one <Seam> is parallel to a plate boundary.

<Feature_relationship> shall not be used to express the ownership of a feature. Each feature shall belong to a part. "Differently from pieces, features are dependent on their bodies, which are called 'hosts'. Features belong to the more general class of parasitic objects like shadows" [Guarino 1997] (see also definitions above). This ownership is not dealt with on the <Feature> level, but is modelled in its subtypes as different requirements may lead to different modelling strategies. Some types of features may require one mandatory parent, as this is the case for <Structural_feature>s. Other subtypes of <Feature> may have several owners and require either a set of parents or are referenced from their owners.

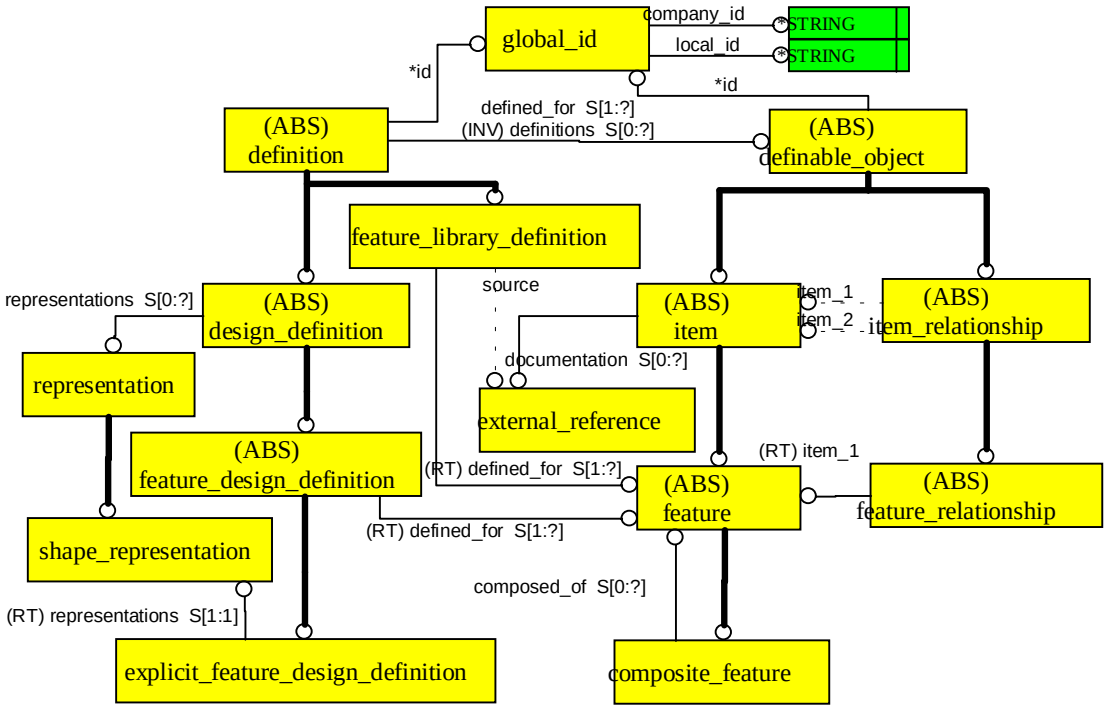


Figure 16: The features model of the Ship Common Model

3.4.2.2 Feature Definitions

All details for a <Feature> are provided by the relevant subtypes of <Definition>. The following definitions are currently available:

- <Feature_design_definition> (abstract)
- <Explicit_feature_design_definition> (subtype of the above one)
- <Feature_library_definition> .

These definitions reference the one or several <Feature>s that they are valid for by means of the <defined_for> attribute which they inherit from entity <Definition>.

3.4.2.2.1 Feature_design_definition

A <Feature_design_definition> is a <Design_definition> and describes the aspects of a <Feature> for the life-cycle phase of design. This <Definition> subtype can not be instantiated. Subtypes are required to describe feature properties, such as geometric parameters, and to restrict the use of the inherited <representations> attribute. Examples of <Feature_design_definition>s for the structural domain are:

- Drain_Hole_Cutout_Design_Definition,
- Round_corner_design_definition,
- Bevelled_corner_design_definition,

- Circular_cutout_design_definition,
- Elongated_oval_cutout_design_definition.

3.4.2.2.2 Explicit_feature_design_definition

The <Explicit_feature_design_definition> describes the design of a <Feature> by an explicit representation of its shape. The shape is given as an explicit geometric model, that is not parametric and without reference to moulded geometry. Only one <Shape_representation> shall be in the set of <representations>. This <Feature_design_definition> is best suited for a purely visual presentation. The functionality or other information of the <Feature> are not included.

3.4.2.2.3 Feature_library_definition

Many types of features, especially form features are standardised. Parameters and properties are collected in for example national standards or in company specific libraries. <Feature_library_definition> describes the location of such a feature description in a library. Library and feature type are identified by names. The location of the library is given as an <External_reference>.

3.4.2.2.4 Positioning of Features

In general <Feature>s shall be positioned using the provisions from the Ship Common Model, that is <Local_co_ordinate_system>. A <Local_co_ordinate_system> is a <Definition> and may, thus, also be applied to <Feature>s. Subtypes of <Feature_design_definition> may, however, define the feature geometry in a way that its position is included. The <Free_form_interior_cutout_design_definition> is an example as the boundary curve of the interior cutout may be represented by a B-spline curve. A <Local_co_ordinate_system> would in such a case be an additional transformation for the <Feature>.

3.4.2.3 Functionality for features

The function of a <Feature> is specified using the existing mechanisms of the Ship Common Model, that is <Functional_definition>. Subtypes of <Functional_definition> shall be introduced to express the role of a <Feature>. Examples of such subtypes are:

- Edge_cutout_functional_definition (access hole, liquid escape, ...),
- Interior_cutout_functional_definition (air escape, penetration, ...).

Feature specific subtypes of <Functional_definition> are not part of the Ship Common Model.

3.4.2.4 Composite features

A <Feature> may be composed of other <Feature>s for the following reasons:

- to model a complex <Feature> that is composed of a set of simple <Feature>s, for example the endcut of a profile.

- to model <Feature>s that are logically defined on an aggregated level like a panel, but physically implemented on the atomic parts constituting the composite object e.g. on the plates realising the panel. A manhole logically defined on a panel can be decomposed into corner-cutouts on the plates that make up the panel. When such decomposition is done, knowledge about the affected connections (e.g. plate seams) is required. The decomposition should first be defined after the plating of the panel is finally determined. Otherwise, if the plating changes, the decomposition structure needs to be updated, too.

A subtype of <Feature> is made a composite feature by instantiating it as a complex subtype, that is combining itself with the <Composite_feature> entity. The <composed_of> attribute of <Composite_feature> references all the <Feature>s that the composite one consists of.

3.4.3 Product Structures

It was found at a number of places within the current shipbuilding APs that similar objects were related to each other and needed to be collected together under certain points of view. Furthermore there is a need to be able to create collections that consist not only of those objects, but also of other collections. The requirement is to create hierarchies of collections of objects and their relationships from different points of view. Examples are:

- The surface of the ship hull that may consist of a number of sub-surfaces (bow shape, bottom shape, parallel midship shape, stern shape, shape of appendices) related by topological and geometric continuity conditions, e.g. the starboard side parallel midship shape is a part of a mathematical plane bounded by (topological relationship) the bow shape, the bottom shape and the stern shape and has a G2 continuity condition (geometrical relationship) to them.
- The steel structure of a ship consists of blocks of groups of design panels of plates and profiles with the geometry derived from the ship interior and moulded form (geometric relationship), which are bounded by the interior and moulded form or by other structural parts (topological relationship) and which are welded together (joint relationship).
- The piping system of a ship consists of the fuel system, the fresh water system, the ventilation system ... where each of them may consist of subsystems of pipes, valves, flanges, pumps ... which are positioned with respect to the steel structure or to the interior or moulded form (topological relationship) and which are connected together and fitted to the steel structure (joint relationship).

In general, each collection of concepts and their relationships, appearing or getting realised (ideally or materially) during the life cycle of the product ship can be modelled as a *Product Structure*. As *Product Structures* are meant to be just views from a certain perspective on the *Ship Product Model* concepts and their relationships can be part of more than one *Product Structure* at the same time.

Common to all *Product Structures* is that there are single objects that are grouped together by several levels of collection hierarchy and that often are related to each other. It seems to be straightforward to model this in a common and generic way and to provide it for reuse.

This is the entry point to the *Product Structure* idea. It is based on the three basic types:

- *Items*, that are the objects of concern (the concepts),

- *Item_relationships*, that set two *Items* into a common context (the relationship between the concepts),
- *Item_structures*, that collect *Items* and *Item_relationships* under a certain point of view (the container for concepts and their relationships).

These three entities are abstract that means not instantiable. However, their functionality to relate concepts and to collect concepts and their relationships can be reused by more special entities, that is by their subtypes.

The ship product model identifies a number of possible “views” of the product. These are

- product structure by system
- product structure by assembly
- product structure by space

These may not be the only “views” possible and the model is flexible enough to allow for others.

3.4.3.1 Product Structure by Space

Product structure by space focuses upon defining spaces such as <Compartment>s and <Zone>s thus dividing the ship. This is commonly known as the ship’s subdivision. The basic principle for <Space>s is that they may contain any number of other <Space>s, thus providing a tree structure or hierarchy of <Compartment>s and <Zone>s.

There are four sets of information to be found within this viewpoint;

- the structures which define the boundary of the space
- the properties associated with that space (e.g. volume, permeability etc.,)
- the relationships between one space and another
- the contents of the space (i.e. pipe, plates, profiles, cableways etc.).

The general purpose or use of this viewpoint is to be able to identify parts of the ship from each <Compartment>, <Space> or <Zone>, as well as it’s relationship to others (e.g. adjacency) and it’s basic properties.

In AP215 (Ship Arrangements), Product structure by space is used as it’s main viewpoint through which it defines subtypes of spaces, identifies cargoes, and allows the assignment of the various cargoes into the different types of spaces defined, permits the establishment of loading conditions (filling of the various compartments), weights and stability information.

3.4.3.1.1 Space_product_structure

A <Space_product_structure> is both a subtype of <Item> and an <Item_structure> represents a collection of <Part>s that are contained within a <Compartment> or <Zone>.

A <Space_product_structure> may be independent of any discipline, or may be defined to consist of <Part>s of any one or more disciplines by including only such parts in the item_relationships.

As <Space_product_structure> is a subtype of <Item_structure>, it inherits a set of <Item>s. These <item>s are restricted at the level of <Space_product_structure> to be of type <Part>.

Therefore, using this model, <part>s such as <pipe>s, <plate>s, machinery or other <space>s may be identified as belonging to the <Space_product_structure>. These <item>s might well be defined outside of this model (perhaps in AP217, AP218 or AP226), and therefore, a set of external defined items are provided for this event.

The <Space_product_structure> itself therefore, acts as a container for those <Item>s. The attribute 'contained_in' references this <Space>.

As <Space_product_structure> is a subtype of <Item_structure>, it inherits a set type attribute 'relationships'. These are restricted at the level of <Space_product_structure> to be of type <Space_product_structure_relationship>. These might also be defined in another model (e.g. AP218) and therefore, a set of externally defined relationships are provided for this use.

Associated with this are the following constraints upon it's usage;

- The <Item>s of a <Space_product_structure> shall be <Parts>. This provides for the items within the space.
- The <Space> associated with <Space_product_structure> through the attribute 'contained_in' shall reference the space which encloses the <Item>s associated with <Space_product_structure> (see above).
- The 'relationships' of a <Space_product_structure> shall be of the type <Space_product_structure_relationship>s.

3.4.3.1.2 Space_product_structure_relationship

A <Space_product_structure_relationship> is a subtype of <Item_relationship> and describes the association of a <Part> and the <Space> with which it is associated. In this manner, it relates those <Part>s which define the <Space> itself (as opposed to those <Part>s which are contained within).

The constraints associated with this are;

- The first item ('item_1') in the relationship shall be the <Space> that is related to a <Part>.
- The second item ('item_2') shall be a type of <Part>.

3.4.3.1.3 Space_product_structure_definition

<Space_product_structure_definition> provides a definition of the design of the <Space>. As it is a subtype of <Design_definition> and <Definition> it inherits a set of <Representation>s which describes the shape geometry of the <Space> it is 'defined_for'. Likewise, each <Definable_object> (e.g. <Space_product_structure>, <Space> and <Space_product_structure_relationship>) may have a set of <Definition>s associated with them, allowing navigation of the model in both directions.

3.4.3.1.4 Space_product_structure_revision

<Space_product_structure_revision> is a subtype of <Revision> which associates a set of <Design_definition>s with the structure discussed above. These <Design_definition>s such as

<Space_product_structure_definition>s allow multiple versions of the product structure, and to specify references to specific versions of the parts contained in it.

3.4.3.1.5 Space

A <Space> is a defined volume on board a ship. A <Space> may be either a <Compartment> or a <Zone>. A <Space> is an <Item> and as such may have a functional definition, design definitions, manufacturing definitions, and product_structure_definitions relating applicable properties to the Space. A geometry defining a <Space> consists of references to the deck and bulkhead moulded form surfaces that bound the space.

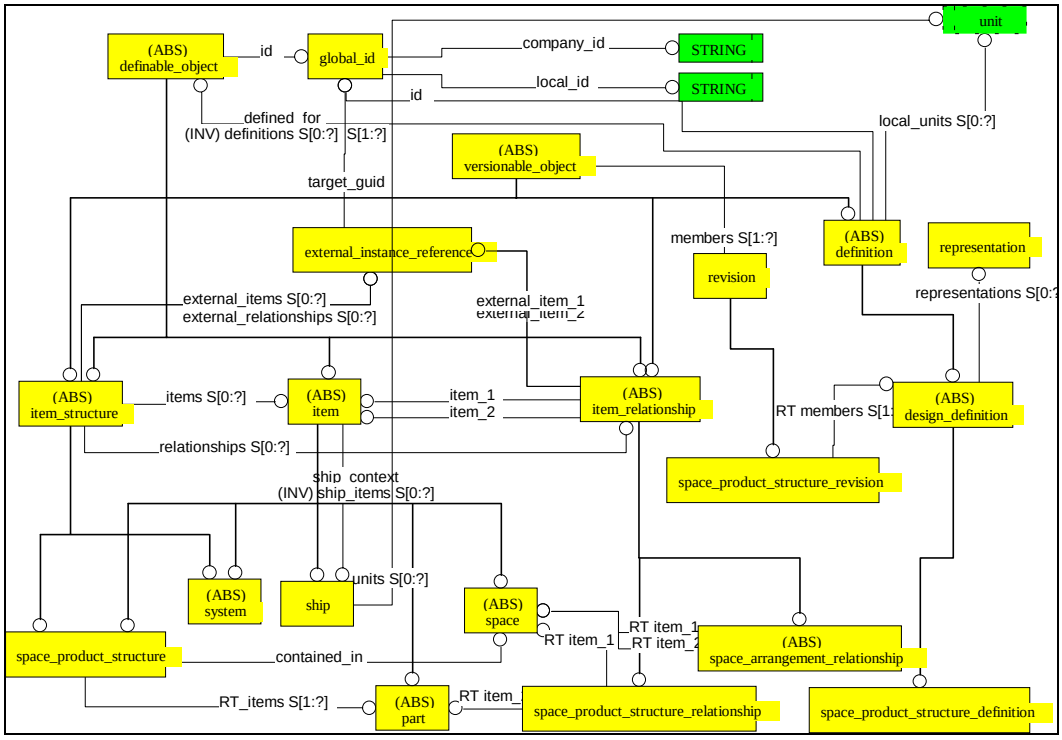


Figure 17: Product Structure by Space

3.4.3.2 Product Structure by System

Product Structure by System is intended to provide the principle of functional views on the *Product Model*. There are already different systems in use by the various shipbuilding APs such as piping system, structural system, machinery system.

A *System* is a function oriented, one-disciplinary view on a group of concepts in a way that either only piping components or only structural components etc. are collected. It allows for a hierarchical, that is tree structure of *Systems*. Therefore *Systems* may consist of other (sub) *Systems*. A *System* is both an *Item* and an *Item_structure*. WHERE rules in *System* ensure that a *System* can only consist of *Parts*, their relationships and other (sub) *Systems*.

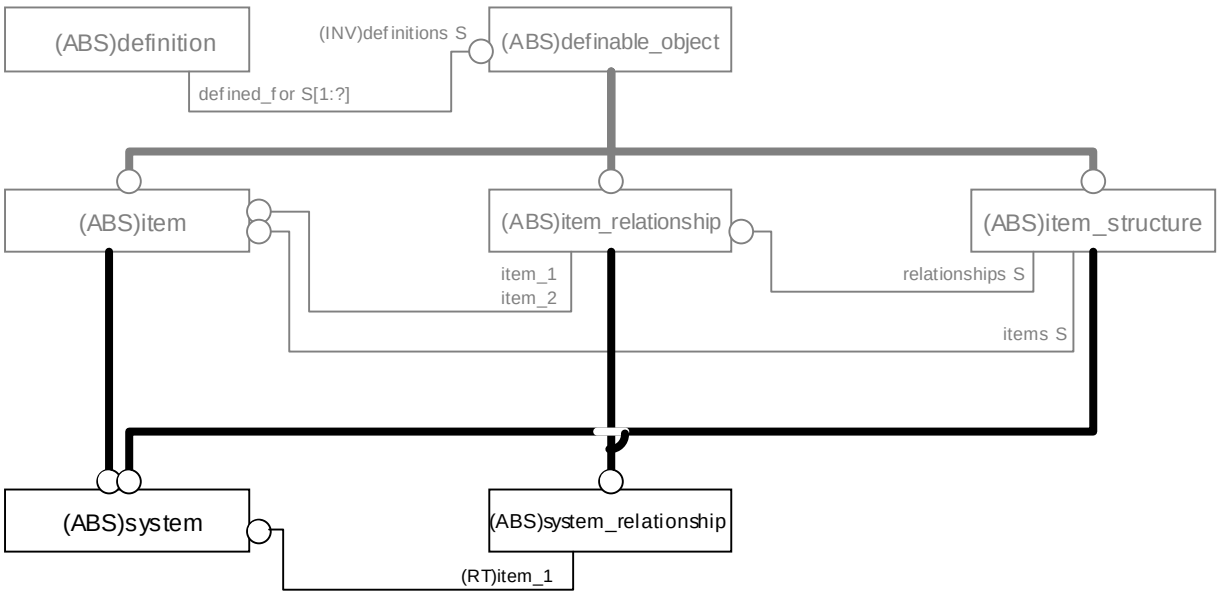


Figure 18: Product structure by system

3.4.3.3 Product Structure by Assembly

Product structure by Assembly specifies how the *Parts* of a ship is made of are collected into greater units from the manufacturing preparation point of view. Such a unit is called an *Assembly*. *Product Structure by Assembly* provides the constructs for defining *Assemblies*. Usually assemblies are cross discipline structures consisting of piping components, structural parts, machinery components, cableways etc. Therefore, *Assembly* is not abstract, but ready for use.

It allows for a hierarchical, that is tree structure of *Assemblies*. *Assemblies* may consist of other (sub) *Assemblies*. An *Assembly* is both an *Item* and an *Item_structure*. WHERE rules in *Assembly* ensure that an *Assembly* can only consist of *Parts*, their relationships and other (sub) *Assemblies*.

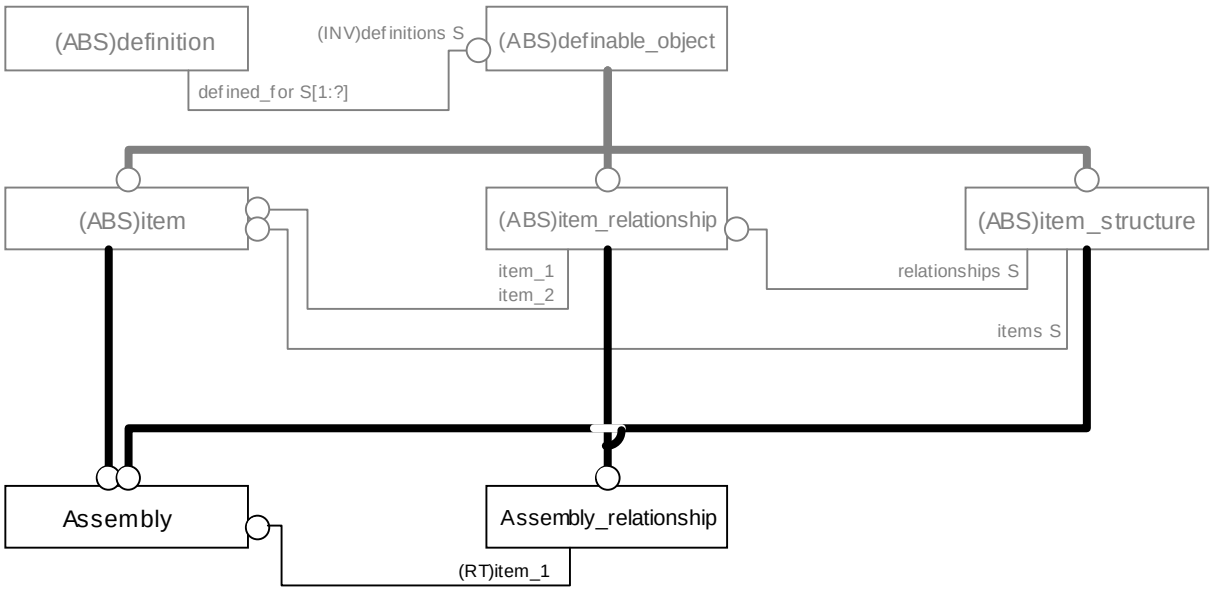


Figure 19: Product structure by assembly

3.5 COMMON UTILITIES

3.5.1 General Characteristics

General characteristics properties constitutes the basic information regarding details of the ship's dimension and identification. This information is independent of any geometric context. It includes scalar values and identification labels for principal dimensions of a ship, designation information for ship related companies, as well as class notation and references to relevant rules and regulations. If this information is not totally consistent with that which can be derived from the ship's geometric representation, then the latter, i.e. the geometrically derived information that shall have precedence.

The main elements provided by general characteristics are:

- Class_and_statutory_designation,
- Class_notation,
- Owner_designation,
- Regulations,
- Ship_designation,
- Shipyard_designation,
- Class_parameters, and
- Principal_characteristics.

A <Class_and_statutory_designation > is a type of <General_characteristic_definition> that specifies the identification given to the ship by the classification society for the purpose of design, manufacture and in service approval.

A <Class_notation> is the collection of information which indicate the classification given to the ship by the classification society.

<Regulation>s provide a set of international and national regulations as well as classification society standards which are used to assess the design, manufacture and in service maintenance of the ship.

An <Owner_designation> is a type of <General_characteristic_definition> that specifies the organisations that own, or are involved with managing, the ship.

A <Ship_designation> is a type of <general_characteristics_definition> that provides an identification given to the ship in order that it can be categorised by any shipping related organisation.

A <Shipyard_designation> is a type of <General_characteristic_definition> that provides an identification given to the ship by the shipbuilder.

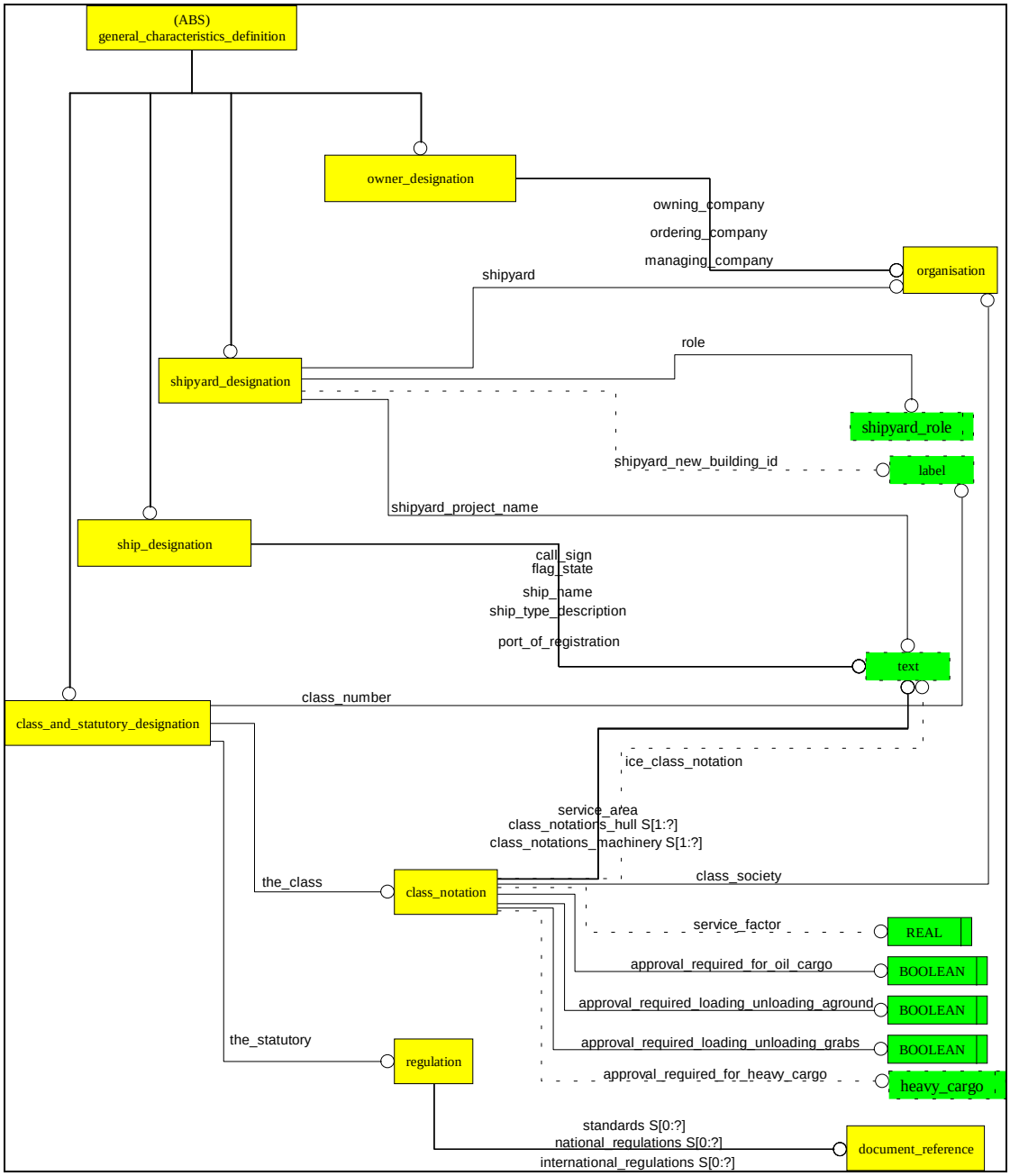


Figure 20: Designation characteristics

<Class_parameters> are a type of <General_characteristic_definition> that specify the length and speed of the ship in accordance with Classification Society rules and statutory regulations.

<Principal_characteristics> are a type of <General_characteristics_definition> that describe the main shape parameters of the hull moulded form. <Principal_characteristics> also includes data that is required in subsequent iterations of the hull development process when one is considering hydrostatics.

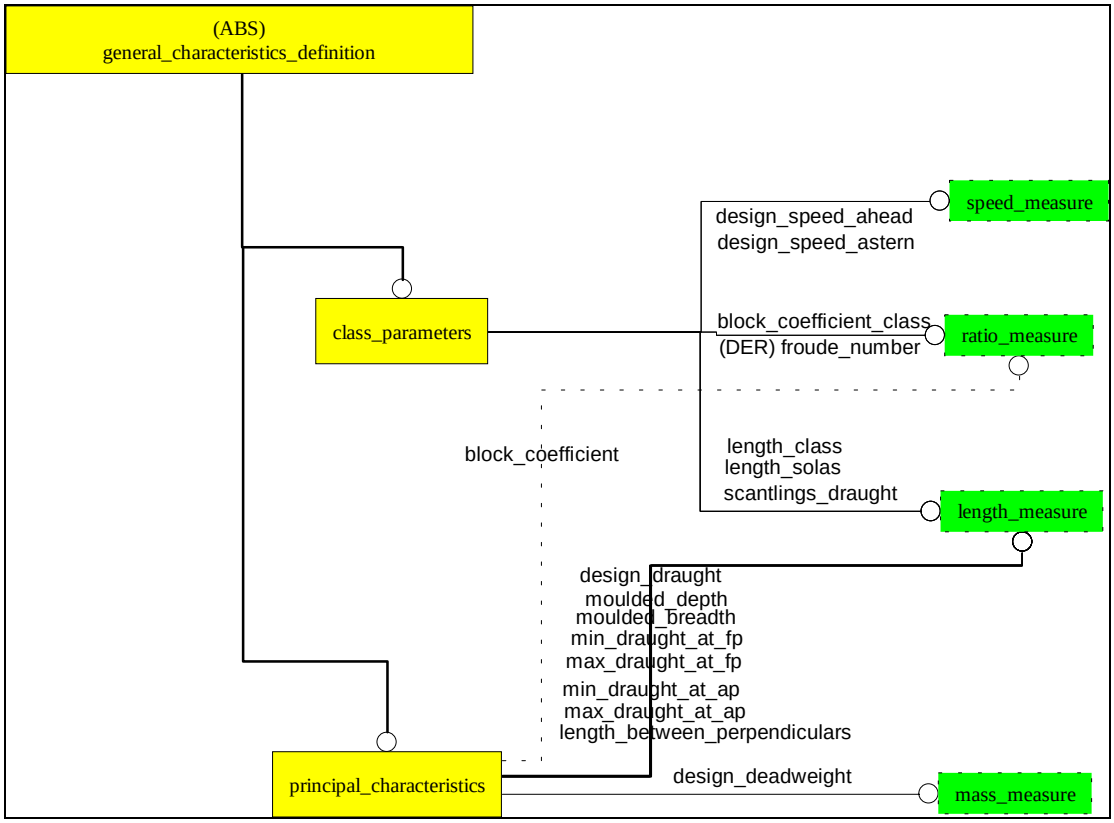


Figure 21: Dimension characteristics

3.5.2 Configuration management

Configuration management within the Ship Common Model includes

- versioning,
- change control, and
- approval.

Those three aspects of configuration management can be used independently of each other. They have in common that they apply to <Definition>s; the versioning, changing and approval of <Definition>s can, thus, be traced within the Ship Common Model. Resources from the Events and Support_resources are applied.

3.5.2.1 Versions

The version schema provides for the versioning of <Definition>s, <Item_relationship>s, and <Item_structure>s; these three constructs and only those are specified to be <Versionable_object>s. The following concepts are supported:

- current version;
- history of versions for the same thing;
- alternatives and merging of versions;
- collections of versions, so called revisions.

The concept of a "version" itself is not provided, that is, there is no entity called version. It is assumed, that any <Definition>, any <Item_relationship>, and any <Item_structure> may play the role of a version. For that reason each instance of those has a <version_id> attribute, which is a human interpretable string. It may be noted that <Item> is not versionable. The reason for this is in the nature of <Item>. Above an <Item> is described as the place holder for a function, for something that lasts throughout the life-cycle of its physical realisation. Once introduced, the "idea" of something can - in the context of the Ship Common Model - only be replaced, but not
versioned.

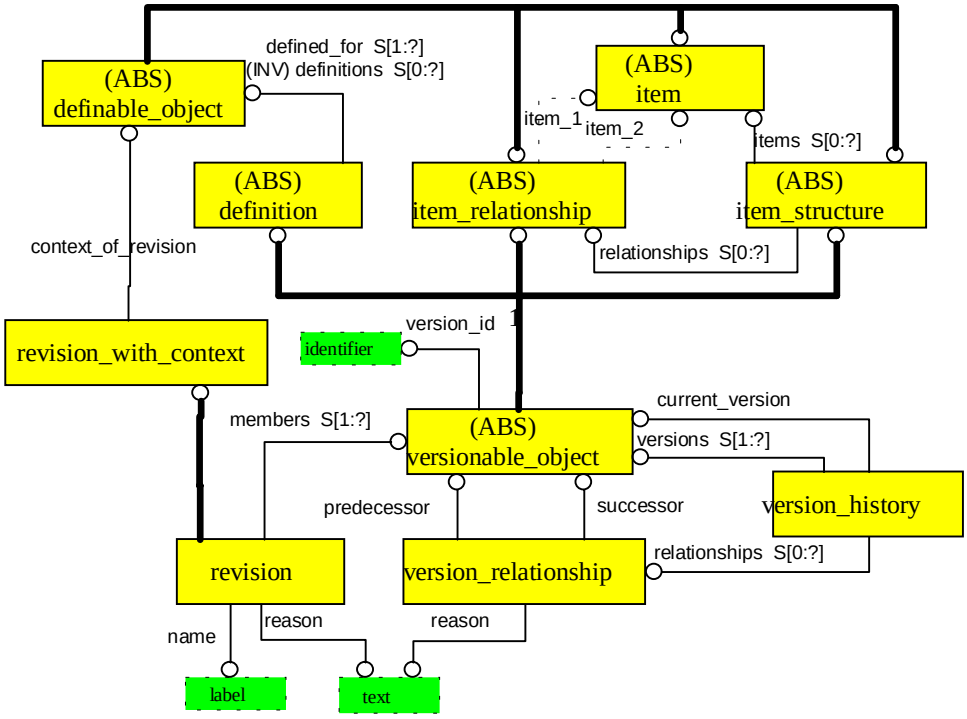


Figure 22: The versions model in the Ship Common Model

3.5.2.1.1 Revision

A <Revision> is a named collection of <Versionable_object>s that belong to each other for a specific <reason>. The <reason> is supplied as a textual description. If information about the person who created the <Revision> and about date and time for this event needs to be included, the subtypes of <Change_event> may be used.

A <Revision> may also be instantiated as a <Revision_with_context>, that is with a reference to an object, i.e. an <Item>, <Item_relationship>, or <Item_structure> that this <Revision> provides details for. For example may all the latest <Definition>s for a plate (design definition, functional definition, manufacturing definition, survey definition etc.) be collected into one <Revision_with_context> that then is assigned to the plate <Item>. In this case it would be useful to add a restriction that for each type of <Definition> there shall be only one instance in the <Revision>.

3.5.2.1.2 Version_relationship

A <Version_relationship> defines the predecessor/successor relationship between two <Versionable_object>s of the same type. Also here, as for <Revision>, the <reason> may be supplied as a textual description and in addition by instantiating a <Change_event> entity.

3.5.2.1.3 Version_history

A <Version_history> collects the different versions of an object, i.e. different alternative descriptions. In addition it specifically identifies the current version. The relationships between the various versions can also be collected in this entity. Thus, a <Version_history> instance may provide the complete history of a <Definition>, an <Item_relationship>, or an <Item_structure>. For example may links to all design definitions of a plate be gathered here with one of those being pointed out as the current design definition.

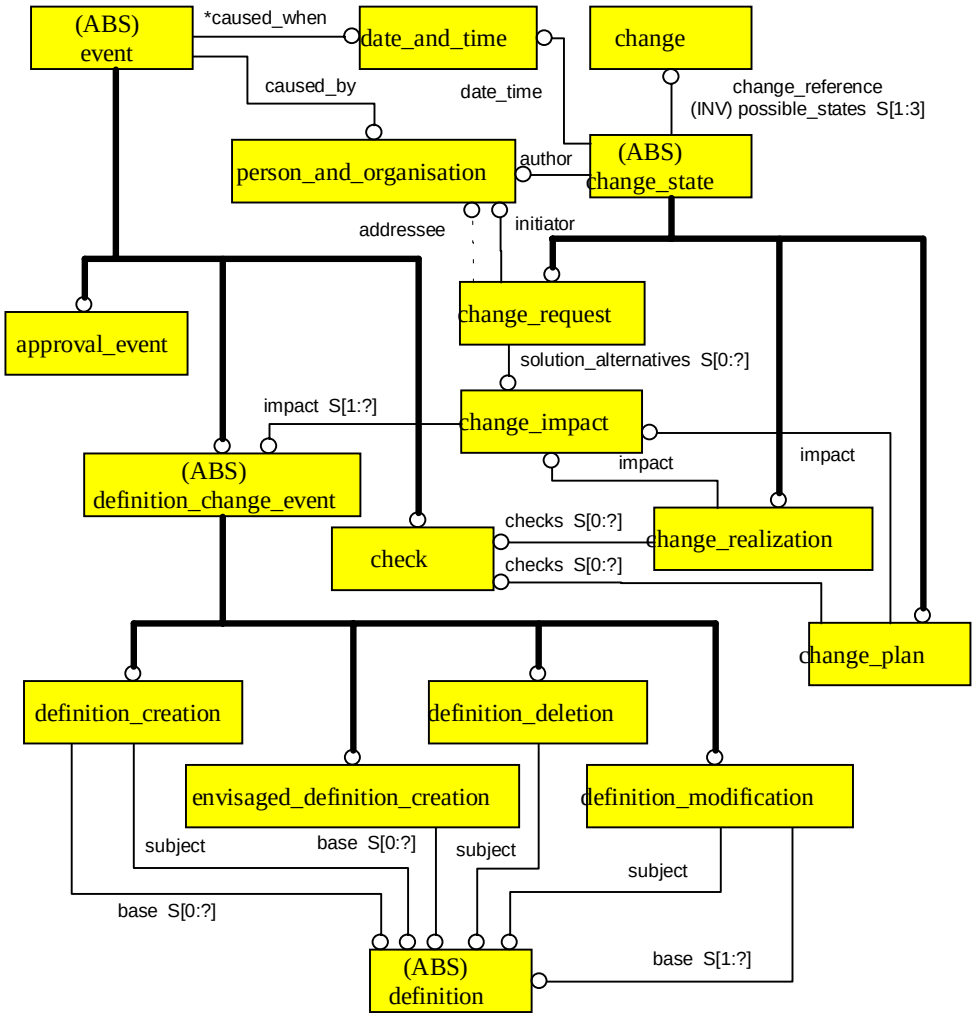


Figure 23: The change model within the Ship Common Model

There are currently no restrictions on the attributes of a <Version_history>. Thus, one or several of the instances in the history log (attribute: versions) may be of a different type than the type of the "current_version". In most cases, however, it is likely to expect them to be of the same nature. The entity specification does not either restrict the <versions> attribute to include or exclude the <current_version>. It may be good practice to include it.

3.5.2.2 Changes

The Ship Common Model allows changes to <Definition>s to be recorded. Changes capture the history, objectives, and administrative details of such changes. The model distinguishes the request for a change, a change plan, and the realisation of a change. All of those <Change_state>s have corresponding data associated with them, such as descriptions of problems and solutions, involved personnel, and planned or performed <Check> events. They all refer to an impact of the change which may be the creation, modification or deletion of one or more <Definition>s.

The concept of a change is independent of the concept of approval. In reality, however, a <Definition> may need to be changed in order to be approved. Such restrictions are not covered by the model.

3.5.2.3 Approvals

The approval concept of the Ship Common Model provides elements for the specification of approval status and approval history of <Definitions>s. It is specified in the approvals. <Approval_history> is the key entity and is specified using entity <Approval_event>. The following approval status are supported:

- noted,
- unapproved,
- conditionally_approved,
- approved,
- rejected,
- user_defined.

More complex approval interrelations may be modelled using approval_constraint_networks. With this it is possible to define the relationship of approval items in terms of the necessary approval of one item in the context of another item. Such a dependency is called "approval relationship". In approval_constraint_networks the dependency is not between two <Definition>s, but between the <Approval_history>s of two <Definition>s; the dependant <Approval_history> can only be approved if also the other <Approval_history> is approved. Available values for the approval status are:

- unapproved,
- approved,
- rejected.

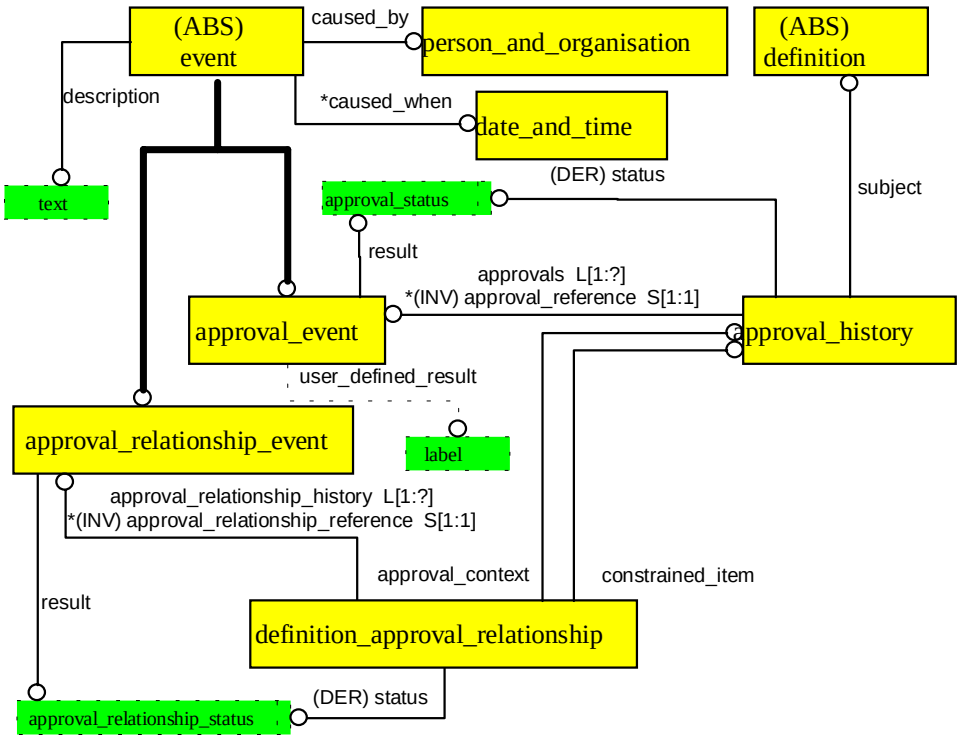


Figure 24: The approval model within the Ship Common Model

3.5.2.3.1 Approval_history

An <Approval_history> collects in chronological order and without redundancy all <Approval_event>s that have occurred up to this point in time for one <Definition>. The current approval status is the <Approval_status> of the last <Approval_event> in the list of approvals. A <Definition> may have zero, one or many associated <Approval_history>s.

3.5.2.3.2 Approval_event

An <Approval_event> triggers the change of an <Approval_status> and results, thus, in one of the above specified status notions. If the status is “user_defined”, the attribute <user_defined_result> shall be provided as the name given to the status. As all <Event>s, also <Approval_event> specifies who is responsible for the approval, when and why did it happen. Each <Approval_event> shall be used by exactly one <Approval_history>; that is, an <Approval_event> can not be used for several purposes. In addition, for all <Approval_event>s the combination of associated <Approval_history> and date and time, when the event occurred, shall be unique; that is, an <Approval_history> may have - and will generally have - several <Approval_event>s, but they shall occur at different times.

3.5.2.3.3 Definition_approval_relationship

A <Definition_approval_relationship> serves the same purpose for approving a relationship as <Approval_history> does for approving a <Definition>. It collects all the <Approval_relationship_event>s and, thus, approval status data in chronological order. This is approval data for an <Approval_history> (attribute: constrained_item) the approval of which depends on the approval of another <Approval_history> (attribute: approval_context).

3.5.2.3.4 Approval_relationship_event

An <Approval_relationship_event> corresponds to the <Approval_event> in the simple case; it notes the change of an <Approval_relationship_status> including the responsible for the change, when and why it happened. The same restrictions apply.

3.5.3 Location Concepts

Within the shipbuilding APs location concepts provides entities for defining a global co-ordinate system, local co-ordinate systems and shipbuilding specific reference systems like spacing tables. They are required for geometric definitions like moulded forms and structural parts, to define the geometric context of the entire ship or any of its components unambiguously in 3D space.

The information covered by location concepts is split into:

- global_axis_characteristics,
- local_coordinate_systems,
- local_coordinate_systems_with_station_reference and
- spacing_grids.

Any co-ordinate system is either a global co-ordinate system or a local co-ordinate system. Additionally spacing tables are used to define positions for iterated parts in the ship. All location concepts are subtypes of *definition*, which means that configuration management like versioning is applicable to them.

3.5.3.1 Global Co-ordinate System

The global co-ordinate system is defined by a *global_axis_placement*. It is unique for **one** instance of ship. All local co-ordinate systems are related to the global co-ordinate system. A *global_axis_placement* defines a fixed system of right handed orthogonal axes to which geometric data are referred (see Figure 25).

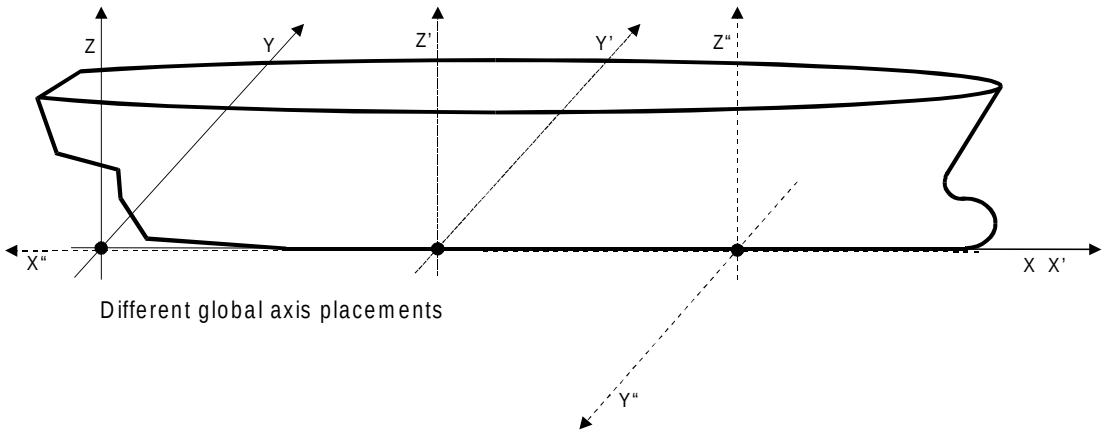


Figure 25: Global axis placements

A *global_axis_placement* is defined by:

- positive z axis in an upwards direction starting from the base of the ship,
- positive x axis running along the ship on the intersection of the centreline with the base and is in one case directed from the after part of the ship to the forward part of the ship or in the other case is directed from the forward part of the ship to the aft part of the ship,
- origin of the global axis placement can be any point on the x axis. The distance of the after perpendicular from the origin and the orientation of the x-axis shall be specified. If any other system of axes is used, local or global, then the transformation relations between it and the global_axis_placement shall be specified.

The data associated with a *global_axis_placement* are the following (see Figure 26):

- after_perpendicular_offset,
- orientation.

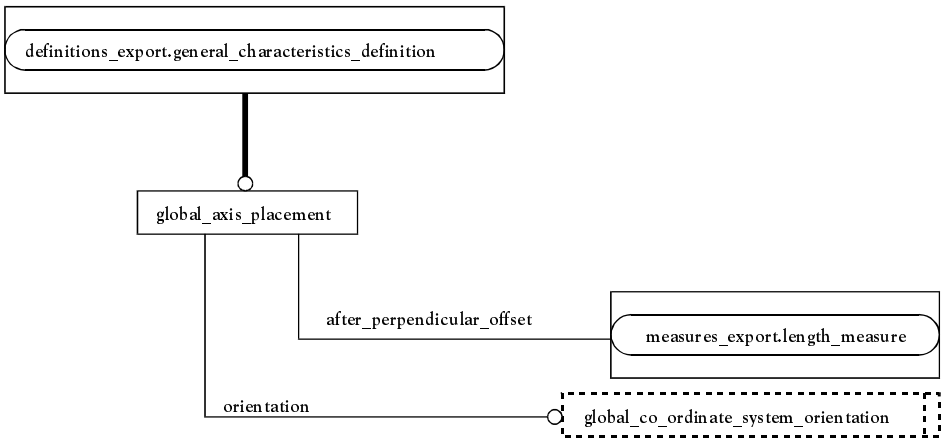


Figure 26: Building Block Global axis characteristics

The after_perpendicular_offset specifies the distance from the origin of the *global_axis_placement* to the after perpendicular.

The orientation specifies the direction of the x-axis and can be one of the following:

- aft_pointing,
- forward_pointing.

Aft_pointing is an orientation of a right handed ship co-ordinate system that has the positive x axis from the forward part of the ship directed to the aft part of the ship.

Forward_pointing instead is an orientation of a right handed ship co-ordinate system that has the positive x axis from the aft part of the ship directed to the forward part of the ship

3.5.3.2 Local Co-ordinate System

A local co-ordinate system is used to define locally the geometric context for components of the ship unambiguously in 3D space. It is defined by *local_co_ordinate_system*. A *local_co_ordinate_system* is a system of right handed orthogonal axes, which is always defined with respect to another co-ordinate system. The relating co-ordinate system might be the global co-ordinate system or another surrounding local co-ordinate system in the same hierarchy tree.

NOTE: Currently a hierarchy of local coordinate systems may cause problems in computation of axes, since the function `build_axes` cannot be redeclared. The `build_axes` function used in *axis2_placement_3d* to calculate the axes of the local co-ordinate system calculates them in global co-ordinates. Alternatively, the attributes axis and/or ref_direction may be redeclared.

As *local_co_ordinate_system* is a subtype of *axis2_placement_3d* the local axes and the origin are handled by *axis2_placement_3d* (see Figure 27 and Figure 28).

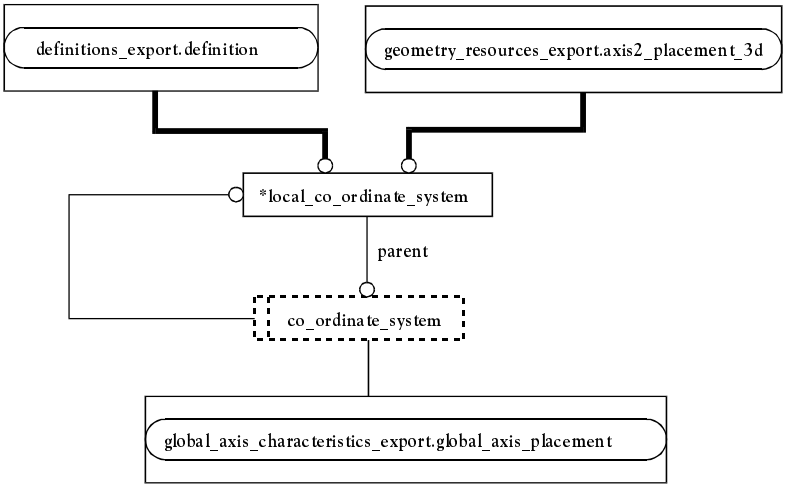


Figure 27: Local co-ordinate system

The data associated with a *Local_co_ordinate_system* are the following:

- parent.

The parent specifies the underlaying related co-ordinate system that serves as definition space for the relating co-ordinate system. An arbitrary number of local co-ordinate systems may exist. Each of them has a parent system, which is either a *local_co_ordinate_system* again or a *global_axis_placement*. Consequently, all co-ordinate systems within a model are structured in a hierarchy. The common root element of this hierarchy is a unique *global_axis_placement*. Therefore geometry data defined in any *local_co_ordinate_system* can be transformed into co-ordinates of the *global_axis_placement*.

The local co-ordinate system represented by an *axis2_placement_3d* from the geometry schema of Part 42 is characterised by a location, an axis and a ref_direction (see Figure 28). The location represents the origin of the local co-ordinate system. Axis and ref_direction are optional and represent the local z-axis and the approximate local x-axis. If one or both of them are not specified the directions are taken from the relating co-ordinate system. Axis is the exact direction of the local z-axis. Ref_direction is used to determine the local x-axis. If necessary an adjustment is made to maintain orthogonality to the axis direction. The p vector with the local x, y and z-axes is derived from the input values or from the related co-ordinate system.

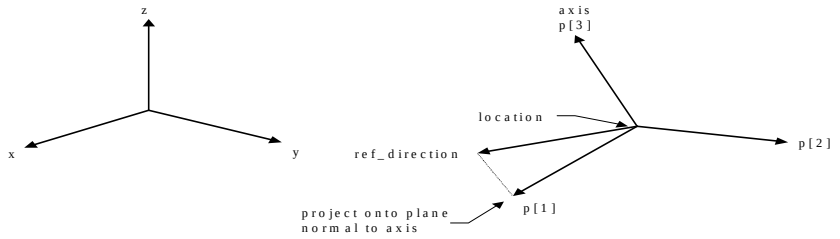


Figure 28: Axis 2 placement 3d

3.5.3.3 Local Co-ordinate System with position reference

A local co-ordinate system with position reference is a specific local co-ordinate system that declares its origin in terms of longitudinal, vertical or transverse positions along the main axes of the ship, e.g. (FR100+50, 400, WL33-100).

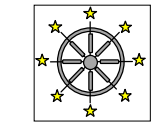
The local co-ordinate system with position reference is defined in reference to the global co-ordinate system of the ship. That implies that the parent co-ordinate system has to be the *global_axis_placement* for the ship. No axis and ref_direction have to be defined as they are derived from the global axes. The location will be derived from the global origin as well. A local co-ordinate system with position reference only moves the origin along the main axes of the global co-ordinate system. There is no rotation for the local axes possible.

The data associated with a *local_co_ordinate_system_with_station_reference* are the following (see Figure 29):

- longitudinal_ref,
- transversal_ref,
- vertical_ref.

A longitudinal_ref specifies either a distance from the global origin on the global x-axis or refers to an existing Longitudinal_position, possibly with an offset value.

A transversal_ref specifies either a distance from the global origin on the global y-axis or refers to a Transversal_position, possibly with an offset value.



A vertical_ref specifies either a distance from the global origin on the global z-axis or refers to a Vertical_position, possibly with an offset value.
All three attributes are optional but at least one has to be instantiated. They are specified either as distance to the origin of the global axis placement by a length_measure or as a *spacing_position*. The spacing position may be specialised to have a certain offset value from a related spacing position. The different types of spacing positions are shown in Figure 31.

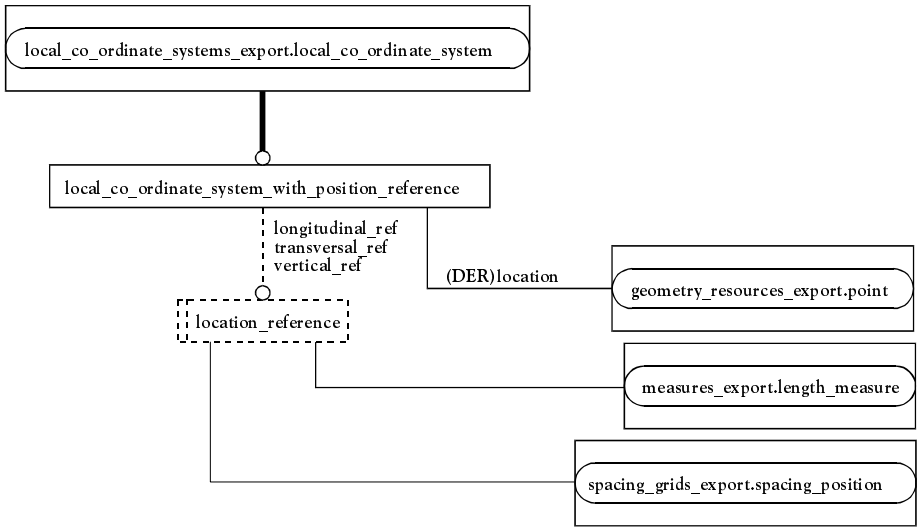


Figure 29: Local Co-ordinate System with position reference

3.5.3.4 Spacing Tables

Spacing tables are specific reference systems used in shipbuilding to define positions for iterated parts in the ship. These positions are called spacing positions, which are defined in the ships global co-ordinate system.

The data associated with a *spacing_table* are the following (see Figure 30)

- description;
- name;
- spacing_table_representations.

The description is optional and specifies the textual account of the reason why the *spacing_table* was created and any additional text that is required to describe the purpose of the *spacing_table*.

The name is also optional specifies the context specific identification for the Spacing_table.

The spacing_table_representations define the positions that make up the table on the co-ordinate axis that is of interest.

A spacing table is a collection of spacing positions that define a list of reference points along one of the global axes of the ship. Specific subtypes of *spacing_table* are introduced. A *longitudinal_table* collects *longitudinal_position*s on the global x-axis. A *transverse_table* collects *transverse_position*s on the global y-axis. A *vertical_table* collects *vertical_position*s on the global z-axis. A *frame_table* is a specific *longitudinal_table* that collects the *longitudinal_position*s of frames. A *station_table* is another specific *longitudinal_table* that collects the *longitudinal_position*s of stations. A *buttock_table* is a specific *transversal_table*

that collects the *transversal_position*s of buttocks. A *waterline_table* is a specific *vertical_table* that collects the *vertical_position*s of waterlines. If there are other types of *spacing_table* required then the general *spacing_table* can be instantiated and the additional information to this spacing table can be stored in the description and name attributes.

Spacing positions can be defined for any of the three global co-ordinate axes of the ship. It is used as a reference point during the design and manufacture of the ship and characterised by identifier and position.

A *spacing_position* is a position on one of the global co-ordinate axes of the ship that is used as a reference point for any geometrical or structural item during the design and manufacture of the ship.

The data associated with a *spacing_position* are the following:

- id,
- location.

The id specifies the unique numerical identification given to the *spacing_position*.

The location specifies the distance of the *spacing_position* from the origin of the global co-ordinate system of the ship.

Several subtypes of *spacing_position* are defined (see Figure 31). A *longitudinal_position* is a *spacing_position* that specifies a location on the x-axis of the global co-ordinate system of the ship. A *transversal_position* is a *spacing_position* that specifies a location on the y-axis of the global co-ordinate system of the ship. A *vertical_position* is a *spacing_position* that specifies a location on the z-axis of the global co-ordinate system of the ship.

A spacing_position_with_offset is a position defined via an offset to an existing spacing position on one of the global co-ordinate axes of the ship.

The data associated with a *spacing_position_with_offset* are the following:

- offset,
- relating_spacing_position.

The offset is the distance to the relating spacing position. The axis, where the distance is measured depends on the type of the relating *spacing_position*.

The relating_spacing_position is the spacing position where the offset is taken from to identify the *spacing_position_with_offset*.

A *spacing_position_with_offset* will be instantiated in a complex instance with a *longitudinal_position*, a *transversal_position* or a *vertical_position*. The following small STEP file shows an example of a *spacing_position_with_offset* instantiation.

```
ISO-10303-21;
HEADER;
FILE_SCHEMA(('SPACING_GRIDS_MODEL'));
ENDSEC;
DATA;
#10 = LONGITUDINAL_POSITION('FR100', 180000.0);
#20 = (LONGITUDINAL_POSITION() SPACING_POSITION('FR100+50mm', *)
SPACING_POSITION_WITH_OFFSET(50.0,#10));
#30 = VERTICAL_POSITION('VFR30', 5000.0);
```

```
#40 = (VERTICAL_POSITION() SPACING_POSITION('VFR30-20mm', *)
SPACING_POSITION_WITH_OFFSET(-20.0,#30));

ENDSEC;

END-ISO-10303-21;
```

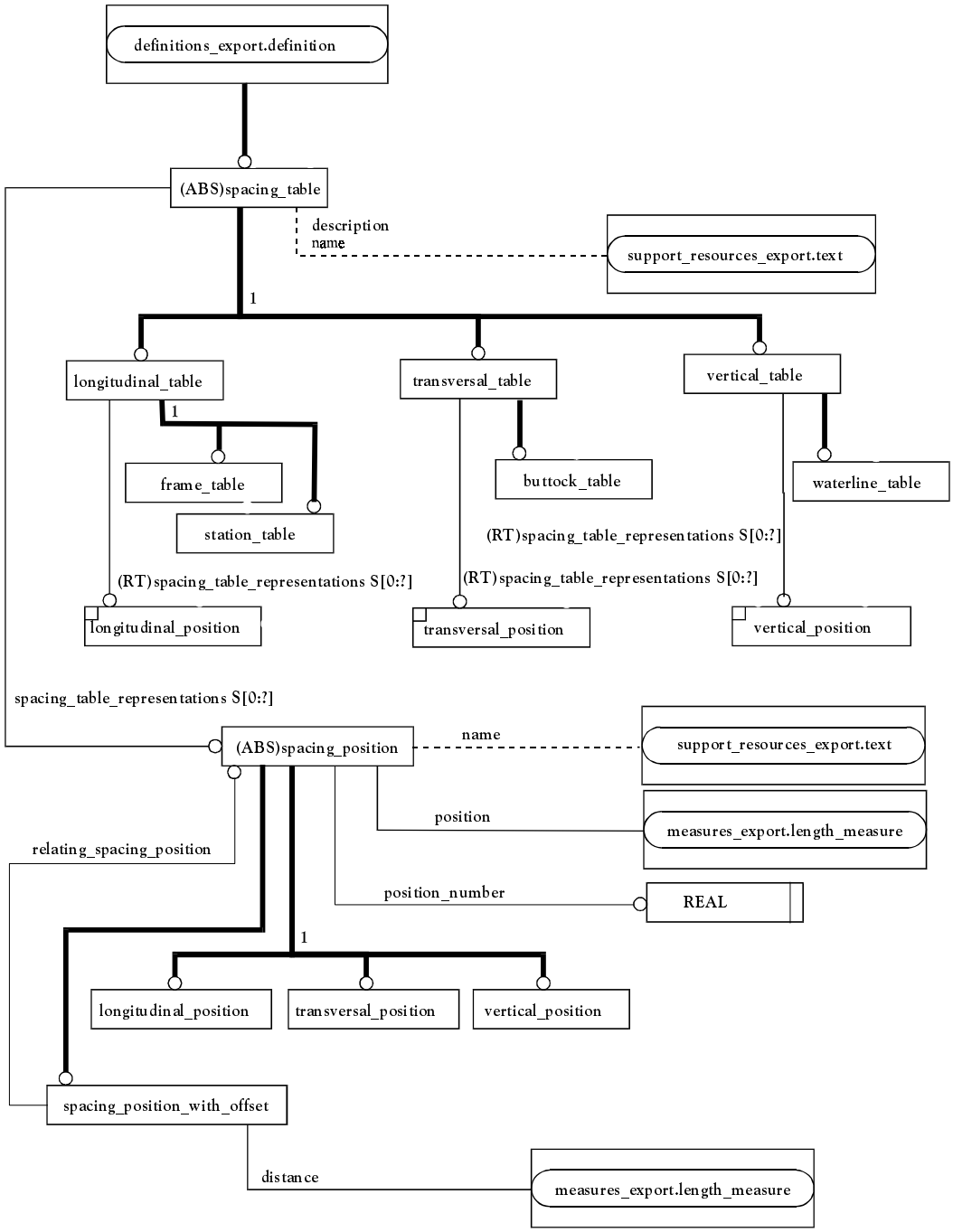


Figure 30: Spacing Grids

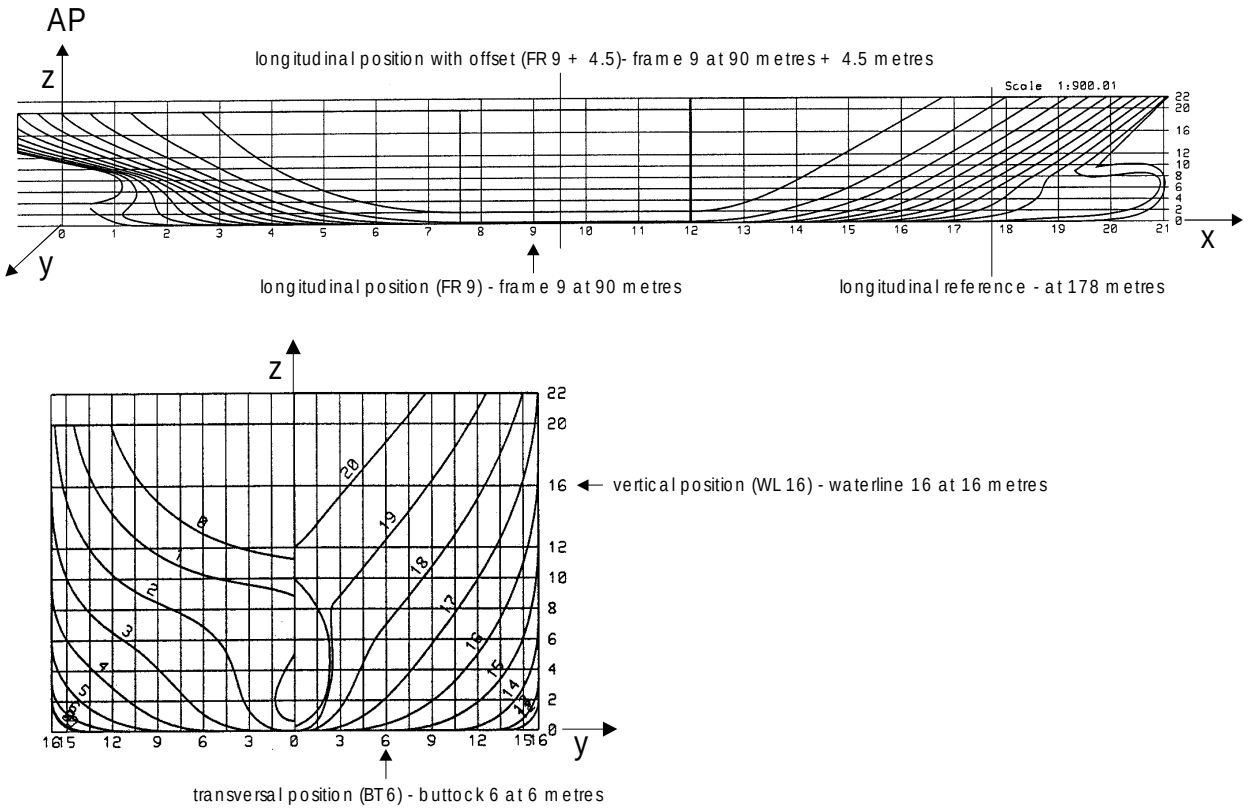


Figure 31: Spacing positions

3.5.4 Moulded Form Geometry

Moulded form geometry provides basic ship geometrical concepts as *ship_point*, *ship_curve* and *ship_surface* which can be used in the shipbuilding AP's. Their intention is to combine geometric information provided by Part 42 with additional shipbuilding specific information.

EXAMPLE: A b-spline curve that can be any 2D or 3D curve has no information about the context in which it will be used. If there is additional information attached to this b-spline curve, for instance that this curve is a waterline, then this information implies that it is a 2D curve in x/y plane of the ship and used in a shipbuilding context. This information can be used by processors to derive additional information.

All three types of ship geometry are subtypes of *representation_item* that allows them to be used in any representation or its subtypes. In AP 216 a *moulded_form_shape_representation* can be instantiated by referencing *ship_point*, *ship_curve* and *ship_surface* to sent for example a list of ship curves together with a surface or wireframe representation of the *moulded_form*.

3.5.4.1 Moulded_form_points

Moulded_form_points introduces *ship_point*, which is a point in a specific context. A Ship_point is a point that is commonly used in naval architecture and that has an associated name (see Figure 32).

EXAMPLE: A ship point may be a knuckle point that has a location in the global co-ordinate system.
The data associated with a *ship_point* are the following:

- point_class,
- point_shape.

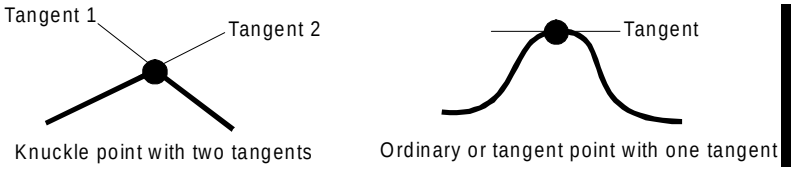


Figure 32: Ship points

The point_class specifies the naval architecture category for the *ship_point*. The categorisation is based on the behaviour of curves passing through the *ship_point*.

The point_class shall be one of the following:

- ordinary,
- tangent,
- knuckle,
- unspecified.

Ordinary is a point of smooth tangency for one or more curves that pass through it. Tangent is a point such that curves passing through it have a specified tangent. Knuckle is a point where a curve passing through the point will have a discontinuous tangent either side of the point. Unspecified is a point where no tangency information is known or recorded.

The point_shape specifies the underlying geometric definition of the *ship_point*.

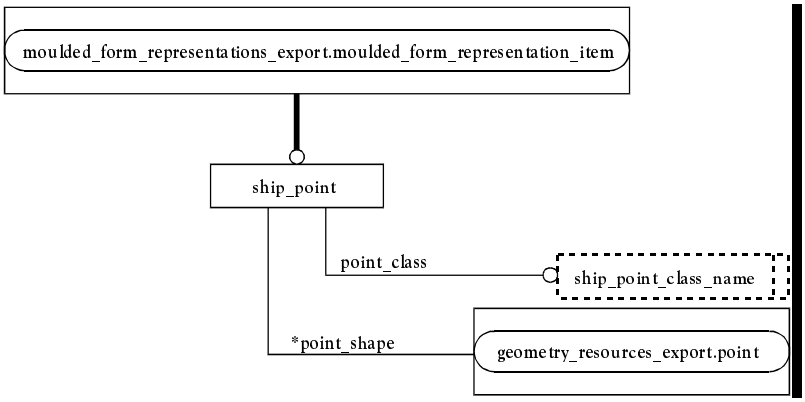


Figure 33: Ship point

3.5.4.2 Moulded_form_lines

Moulded_form_lines introduces *ship_curve*, which is a curve in a specific context. A *ship_curve* is a curve that is commonly used in naval architecture and that has an associated name.

EXAMPLE: A waterline that has geometry defined by a b-spline curve.
The data associated with a *ship_curve* are the following (see Figure 37)

- curve_class,
- curve_shape,
- side_condition.

The curve_class specifies the naval architectural category for the *ship_curve*. The categorisation uses curves that are commonly required for the design definition of the Moulded_form.

The curve_class shall be one of the following:

- buttock_line,
- centreline,
- deck_line,
- flat_of_bottom,
- flat_of_side,
- intersection_line,
- station_line,
- waterline,
- bounding_line,
- unspecified.

A buttock_line is a curve that is the intersection of a longitudinal plane with a hull moulded form (see Figure 34)

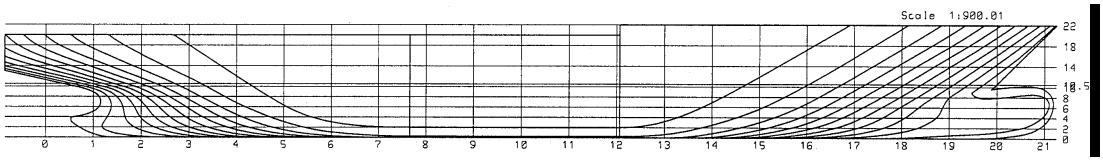


Figure 34: Buttock lines

A centreline is a curve that is the intersection of the longitudinal centreplane with the hull moulded form.

A deck_line is a curve lying on the moulded surface of a deck.

A flat_of_bottom is a curve, which is the boundary of the planar surface of the bottom at the base of a hull moulded form.

A flat_of_side is a curve, which is the boundary of the planar surface of the side at the outer-most port, or starboard side of a hull moulded form.

An intersection_line is a curve that is the intersection of two surfaces found on or within a moulded_form

A station_line is a curve that is the intersection of a transverse plane with a hull moulded form (see Figure 35).

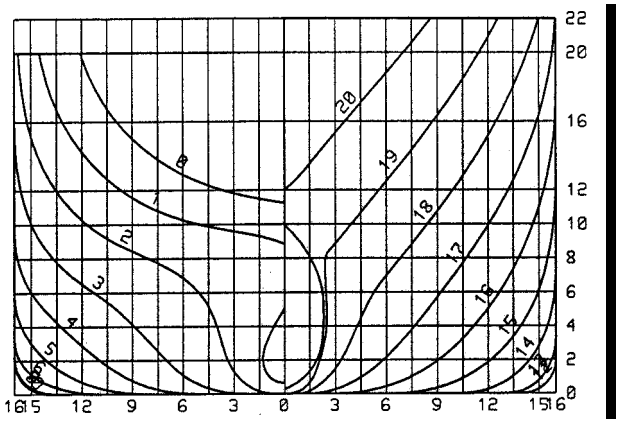


Figure 35: Station lines

A waterline is a curve that is the intersection of the water plane with a hull moulded form (see Figure 36)

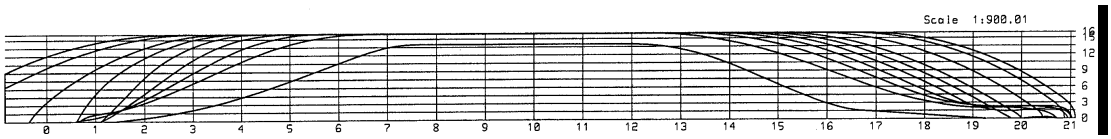


Figure 36: Waterlines

A bounding_line is any bounding curve of a *ship_surface*.

An unspecified curve is a line whose relation to naval architecture is not known or not recorded

The curve_shape specifies the underlying geometric definition of the *ship_curve*.

The side_condition specifies the behaviour of curves crossing the defined curve at the cross over point.

EXAMPLE: A station line will knuckle at a waterline representing the intersection of the bilge keel and the flat of side.

The side_condition shall be one of the following:

- knuckle,
- smooth,
- tangent,
- unspecified.

The side_condition is knuckle when the curve passing across the curve will have a discontinuous tangent either side of the crossover point.

The side_condition is smooth when the curve passing across the curve will be smooth.

The side_condition is tangent when the curve passing across the curve has a specified tangent.

The side_condition is unspecified when the curve passing across the curve has no tangency information.

A ship_curve_with_spacing_position is a ship curve with additional naval architectural meaning, which has an associated spacing position.

EXAMPLE: A waterline that has geometry defined by a b-spline curve and located at a vertical spacing position.
The data associated with a *ship_curve_with_spacing_position* are the following (see Figure 37:

- location.

The location specifies the position in the global co-ordinate system, where the ship curve is defined. The ship curve has to be a 2D curve.

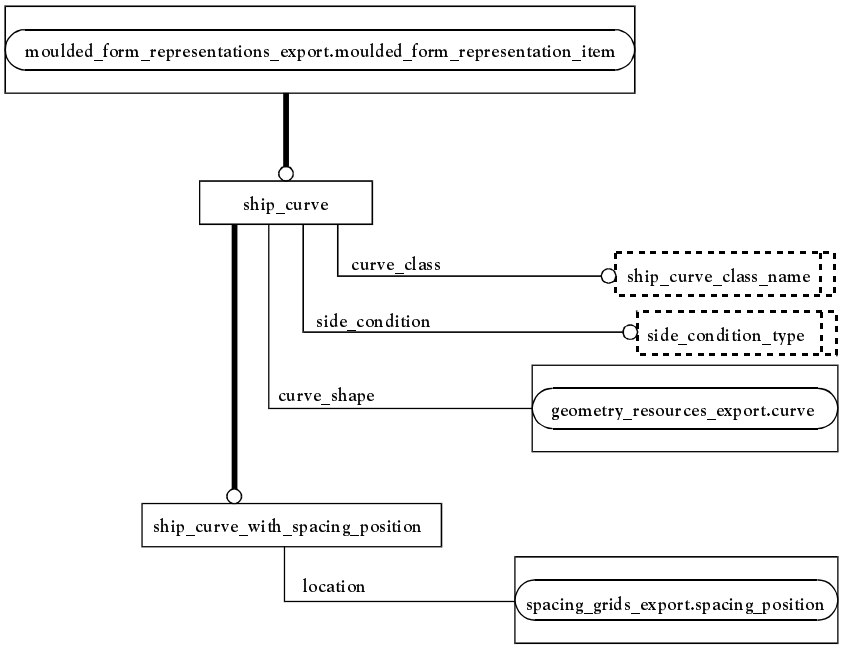


Figure 37: Ship curve

3.5.4.3 Moulded_form_surfaces

Moulded_form_surfaces introduce *ship_surface,* which is a surface in a specific context. A *ship_surface* is a surface that is commonly used in naval architecture and that has an associated name.

The data associated with a *ship_surface* are the following (see Figure 38)

- surface_class,
- surface_shape.

The surface_class specifies the naval architectural category for the *ship_surface*. The categorisation is based on the distinction between the location of the surface.

The surface_class shall be one of the following:

- external_surface,
- internal_surface,
- blending_surface.

An external_surface is a surface that represents geometrically the outside part of a component of the ship.

EXAMPLE: An external surface is the flat of side of the ship hull.
An internal_surface is a surface that represents geometrically internal parts of a component of the ship.

EXAMPLE: An internal surface is a thruster tunnel of the ship hull.
A blending_surface is a surface that represents geometrically a connection between different parts of a component of the ship.

EXAMPLE: A blending surface is the connection of the hull and the bulbous bow of the ship hull.
The surface_shape specifies the underlying geometric definition of the *ship_surface*.

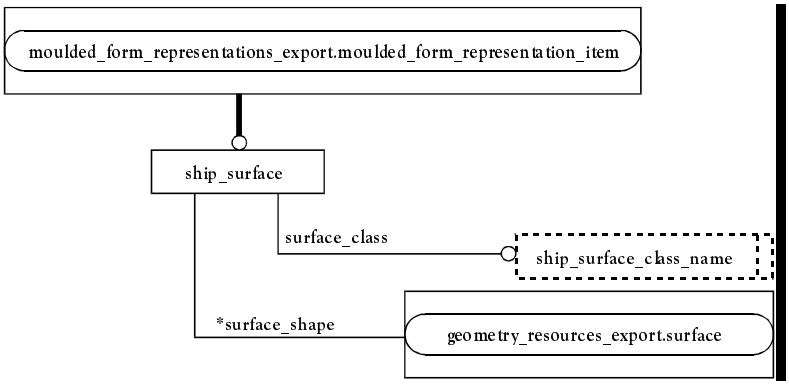


Figure 38: Ship surface

3.5.5 Ships

The context for all shipbuilding APs is the ship. It is the unique root entity for the ship product model and therefore root entity for all shipbuilding APs. All information created by any shipbuilding AP is related to *ship*. The information covers the entire lifecycle of the ship. At the beginning of the lifecycle only a few parameter are assigned to the ship. During the lifecycle lots of additional information will be added to the ship.

The data associated with a *ship* are the following (see Figure 39):

- units.

The units define a set of pre-defined units for the entire ship product model. In addition for each definition local units can be defined. These local units then overwrite the global units.

EXAMPLE: Meter can be defined for all length measures.
The link between the ship entity and all other information is established via the context attribute of item that has to be instantiated for all subtypes of item excluding the ship. To find all information to a particular ship the item subtypes that have this ship as their context have to be collected. Then all the definitions that are pointing to these items can be collected.

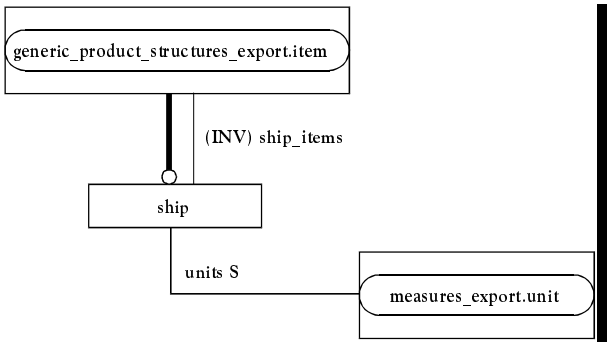


Figure 39: Ships

The specific types (see Figure 40) of the ship are handled by ship_types where functional definitions for commercial and naval ships are defined and pointing to *ship*.

Figure 40: Ship types

3.5.6 Ship Materials

Ship Materials is used to specify the raw materials used to manufacture the ship. Depending on the requirements on the information about the material there are several ways to describe them:

- <Ship_material> - identify a raw material in general by a (possibly company/system specific) name
- <Homogeneous_ship_material> - identify a raw material by it's physical properties (youngs_module, yield_point, stress_of_fracture, poisson_ratio etc.)
- <Weld_filler_material> - identify a raw material for welds by it's chemical composition (carbon, silicon, manganese, phosphorus, sulphur etc.)

Specifying a material by its physical properties removes the dependency upon trade names which may changes over time and from region to region.

Being <Definition> subtypes <Ship_material>s are getting assigned to the <Item>s they are defining properties for. Everything made of a material - e.g. every <Part>- shall have its material data defined using this *Common Utility* at some point in the life cycle.

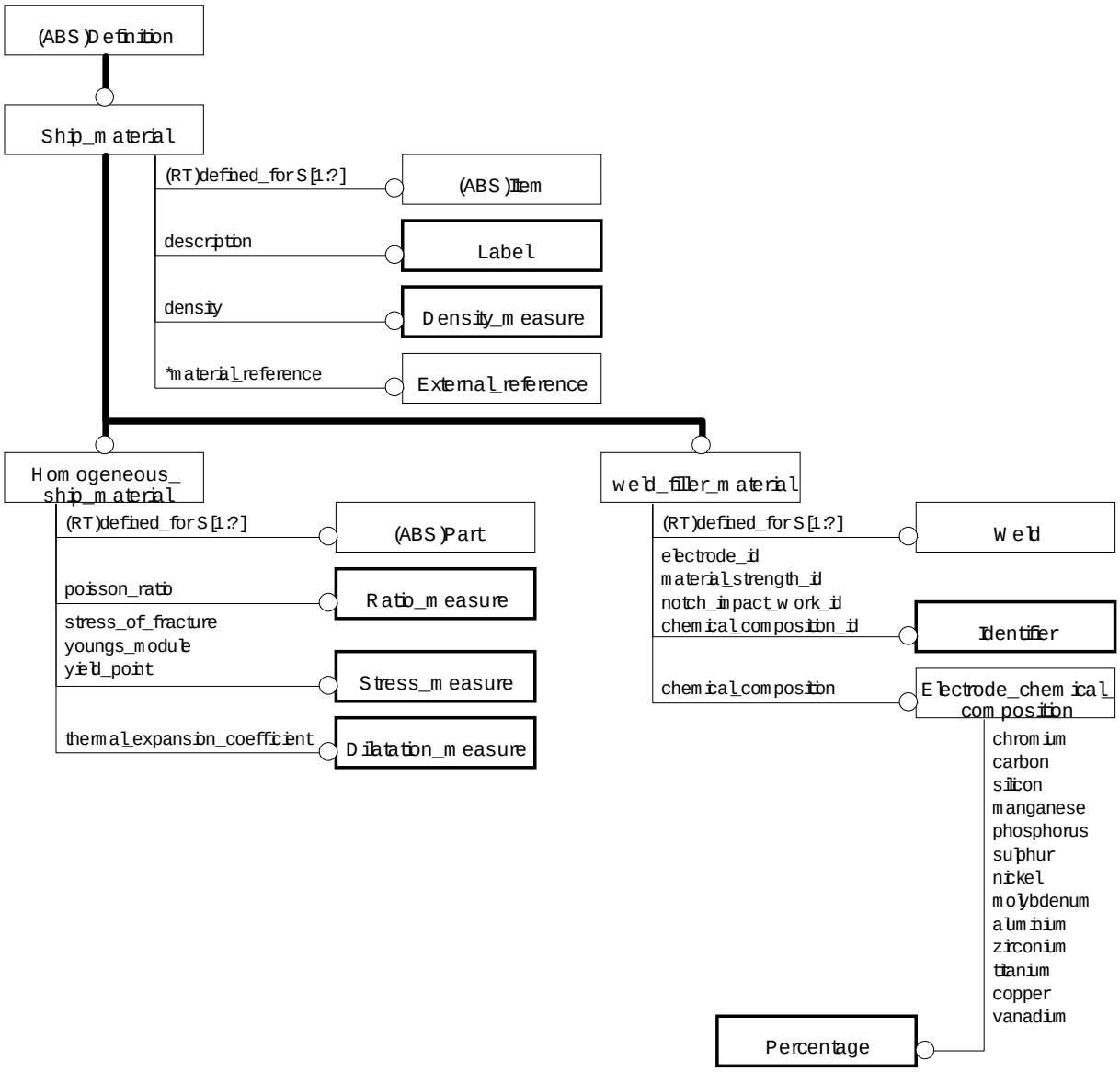


Figure 41: Materials

3.5.8 Externally Defined References

Externally Defined References is a *Common Utility* that provides a mechanism to refer to data that is outside of the scope of a certain data exchange. This data might either be a - printed or digital - document that is refereed to, an entity instance that is part of another data exchange or a reference into a parts library. This capability allows to exchange data very efficient because data that has already been exchanged or data that is available from other sources need not to be part of the transferred.

3.5.8.1 External References

External References are intended to refer to a document as a source for further information about a concept or its properties. Even if the mechanism provided does not give information about the internal structure of the referred document it provides the necessary information to access it. This might be a uniform resource locator in case of a digital document that allows a receiving application to directly get access the information source and show it in a suitable viewer.

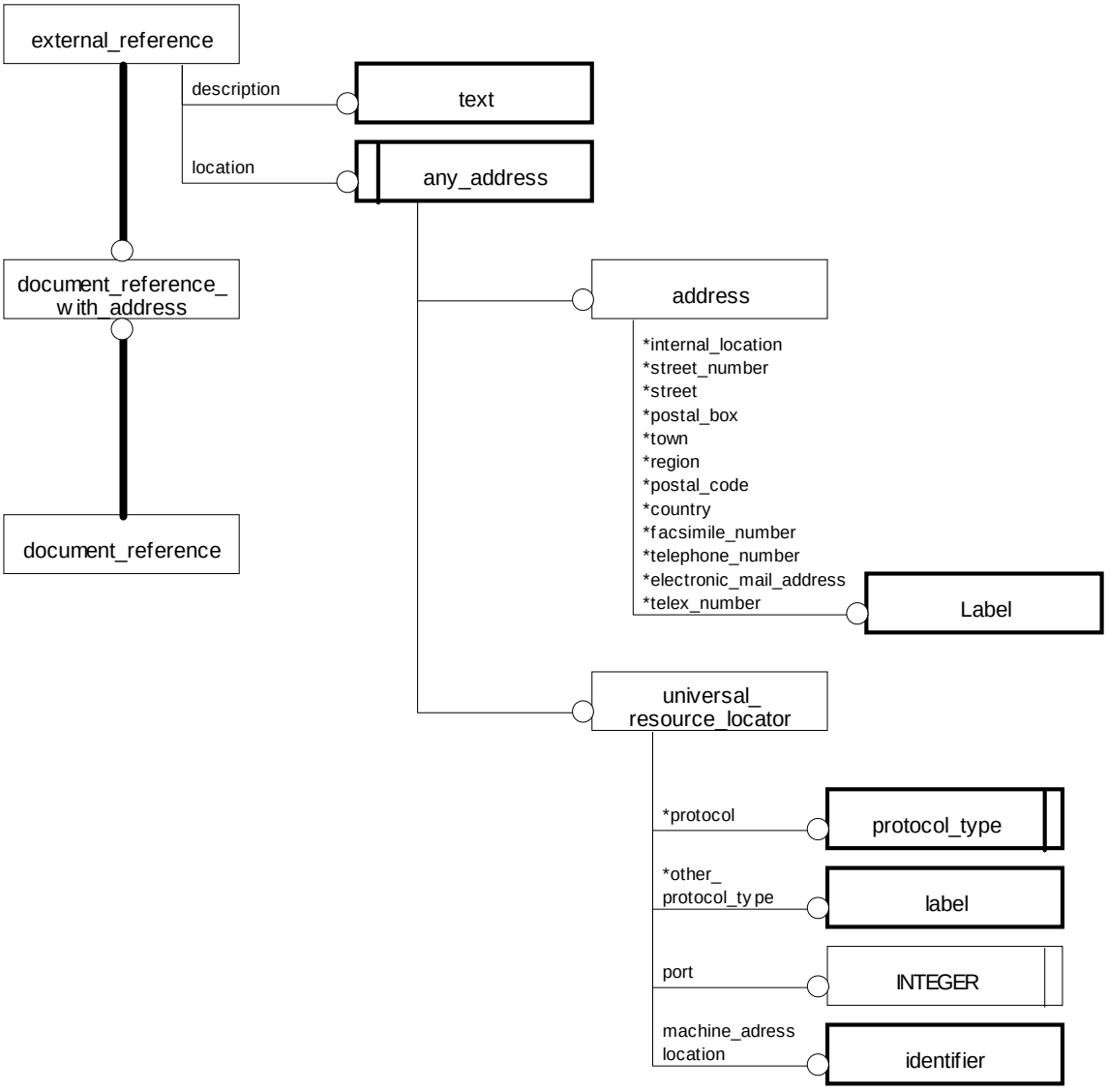


Figure 43: External Reference

3.5.8.2 External Instance References

External Instance References satisfies the need within the shipbuilding APs (but not only there) to allow references between instances across data exchange boundaries for the following reasons:

- to preserve relationships between concepts or their properties that are defined in different APs (e.g., a structural penetration defined in AP218 may reference the pipe defined in AP217 which passes through the penetration, a plate defined in AP218 may reference a moulded hull form defined in AP 216 defining the geometry of a hull or bulkhead).
- to preserve relationships between instances which are defined in the same AP, but are exchanged in separate data transfers (e.g., a pipe in one data transfer may be connected to a pipe which was transferred in another exchange).

External Instance References are used to provide a placeholder for the destination of a 'has a' relationship between two concepts or their properties for the case that their instances are not part of the same data exchange.

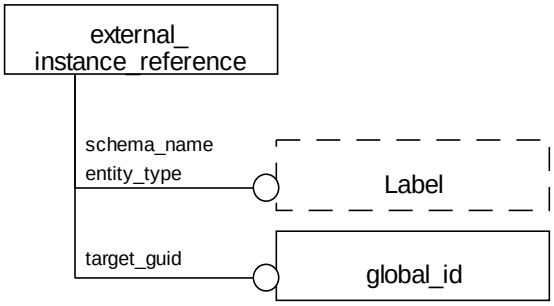


Figure 44: External instance reference

3.5.8.3 Library element references

ISO 13584 "Parts library" describes a general mechanism to build up libraries of parts. This mechanism consists of four different services that provide different levels of detail for the description of the library parts:

- Service 1: capability to express that a classification of a part etc. is defined in a PLIB-compliant library
- Service 2: capability to express that a property of a part etc. is defined in a PLIB-compliant library
- Service 3: capability to express that a part, etc. itself is defined in a PLIB-compliant library, i.e., as a PLIB-catalogue-defined part
- Service 4: capability to express that the (parametric) representation (of a part) is defined in a PLIB-compliant library

For the ship product model service 3 was identified to best fit into the requirements:

- designation of a class, the part is an instance of,

- a set of pairs of (property BSU, property value), and
- an association which expresses, that the above information identifies the part (i.e., a definitional association).

This has been taken as the basis for Plib_reference which is the ship product model way of referencing parts in a library.

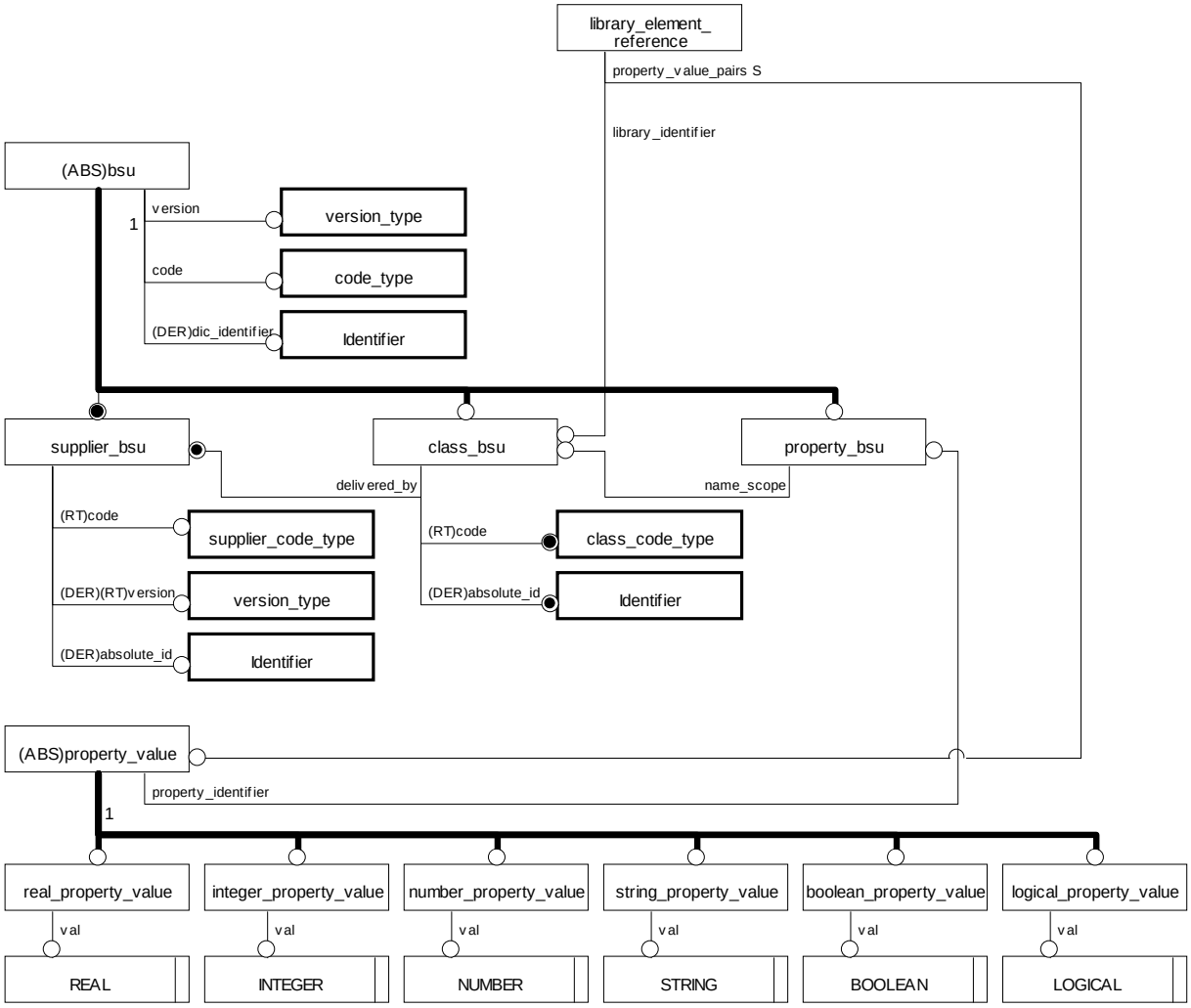


Figure 45: Library element references

4 AP Modularization

4.2 Building Block Approach

An Application Reference Model in the context of the ISO/STEP methodology is the representation of application domain specific product model requirements. The shipbuilding community is developing several such reference models, each to provide the requirements for a distinct shipbuilding Application Protocol. However, these reference models overlap with respect to the concepts found in the shipbuilding domain. The intention is to

- make such overlaps explicit through use of identical Application Reference Model subsets;
- to provide for a more efficient Application Reference Model development through the reuse of elements (potentially) common to two or more Application Reference Models.

The principle means to support this is the concept of *Building Blocks*. A Building Block is a generic EXPRESS-based construct for the confined representation of a Unit of Functionality in part or in whole. Therefore, a Unit of Functionality may be represented by a single Building Block or many.

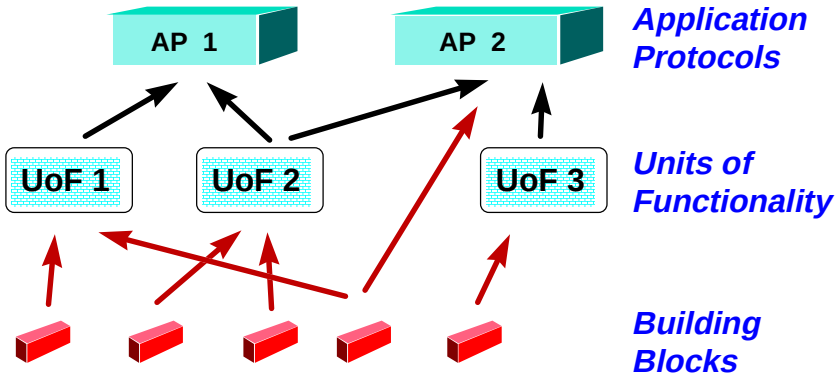


Figure 46: Building Block Approach

The Building Block concept and other elements introduced in this document do *not* exist in the context of the development of Application Protocols in accordance with the ISO/STEP methodology. It is however easily possible to transform a set of building blocks into an Application Reference Model as required by the ISO/STEP methodology. This process is known as longform generation and the functionality is provided by the usual EXPRESS toolkits.

A similar approach is also used to semi-automatically generate the *documentation* of the ARM for an AP in the shipbuilding group, and to ensure this, a set of guidelines are set up to describe all the details necessary to conform to the approach.

4.2.1 The Building Block Structure

A *Building Block* is an EXPRESS-based specification that is used to define Units of Functionality. A Unit of Functionality may include one or several Building Blocks. A Building Block consists of three schemas,

- an *import* schema providing an interface to those elements of other Building Blocks that are used in the model schema of this Building Block
- a *model* schema, carrying the data structure of the Building Block; all types that are used in the model schema are either defined locally or referenced from the belonging import schema via USE FROM, and
- an *export* schema, making available those elements defined in the model schema intended to be used by other Building Blocks.

As the modularity concept as such is not well supported by EXPRESS, a number of restrictions and agreements apply to the EXPRESS usage. They are related to

- the use of OPTIONAL attributes
- the use of ENUMERATIONS
- the use of ANDOR resp. ONEOF
- the use of STRING types
- the use of SELECT types

4.2.1.1 Building Block Syntax – Example

```
(*
Building Block Name: structural_parts
Editor: Thomas Koch (KCS)
E-mail: tk@kcs.se
Version: $Revision: 1.2 $
Status: $State: Draft $
Last Edit: $Date: 1993/09/30 13:00:12 $
Description: A structural part contains the properties common to
all elements of a structural system.
Open Issues:
Rationale: The structural part BB is created based on requirements
from AP218.
*)
```

The above header allows some automatic processing by the Building Block e-mail server.

```
SCHEMA example_import;
    USE FROM ship_parts_export(ship_part);
    USE FROM global_reference_system(location_on_mould_line);
END_SCHEMA;
```

The import schema describes all necessary links to other Building Blocks.

```
SCHEMA example_export;
    USE FROM example_model(structural_part);
```

```
END_SCHEMA;
```

The export schema makes those elements of the model schema (below) public which may then be used by other Building Blocks.

```
SCHEMA example_model;
    ENTITY structural_part
      ABSTRACT SUPERTYPE
      SUBTYPE OF (ship_part);
        location: location_on_mould_line;
        (*...*)
    END_ENTITY; (* structural_part *)
END_SCHEMA;
```

4.3 ISO SC4 MODULARIZATION APPROACH

The traditional mechanism for STEP AP development is seen to have a number of drawbacks. These range from the high cost of developing an AP in terms of the length of time required, the expectation from vendors for the reuse of application software, to the duplication and repeated documentation of the same requirements in different APs. Other observations include the fact that AP interoperability is flawed, especially when it comes to reusing data generated by an implementation of one or more APs by a different implementation using one or more different APs. Then there are the difficulties experienced by companies requiring the implementation of a combination of multiples APs or AP extensions. Thus the traditional approach of STEP results in many islands of APs which exist autonomously.

4.3.1 Problems with the Building Block Approach

The Shipbuilding Team has addressed some of these of these issues through the adoption of the Building Block approach as a distributed and concurrent development methodology where small units of the model are constructed to allow their reuse into Units of Functionality.

Although this overcomes a number of the problems outlined above, it does not deal with how those entities in the Application Requirements Model (ARM) are interpreted in the Application Interpreted Model, and has, to date, only been used with respect to ARM development.

The AIM is the result of interpreting the Integrated Resources (IR’s) to fulfil the requirements described in the ARM. The theory is that because each AP uses a standard set of common resources in the AIM, then the APs will be interoperable over those resources when implemented.

However, differences in the way in which interpretation is carried out means that interoperability is not guaranteed and in most cases of today’s APs interoperability does not work because of these differences.

Having stated this, most APs use geometry to some extent and this was seen as one area in which a common interpretation could be reused across the various APs through the use of Application Interpreted Constructs.

4.3.2 Application Interpreted Constructs

The development of the Application Interpreted Constructs (AICs) went some way to address some of the problems pointed out above. Since the actual ISO standard which is implemented is the EXPRESS of the AIM and not the ARM, and with each AP being interpreted subtly different each time, meant that the Integrated Resources (or Part 40's) were being used inconsistently across the various APs. AICs provided a consistent "view" or common interpretation for the use of geometry for many APs.

This was in some ways a natural choice since most[1] APs will need at some stage to use geometry to describe their shape. Thus, the AICs ensured interoperability at the geometrical level.

However, these only cover the geometrical requirements of the APs, which still need to be described in the ARM along with rest of the model.

Thus some mechanism in STEP is required which combines the modular development methods of the Shipbuilding Group with that of the AIC usage.

4.3.3 AP Modularity

For these reasons members of the STEP/SC4 team within ISO have been looking at the benefits of a more modular approach (similar to that of the Shipbuilding Team) to overcome these problems. Figure 47: The move towards Modules shows how Modules are expected to become the development mechanism by the 21st Century.

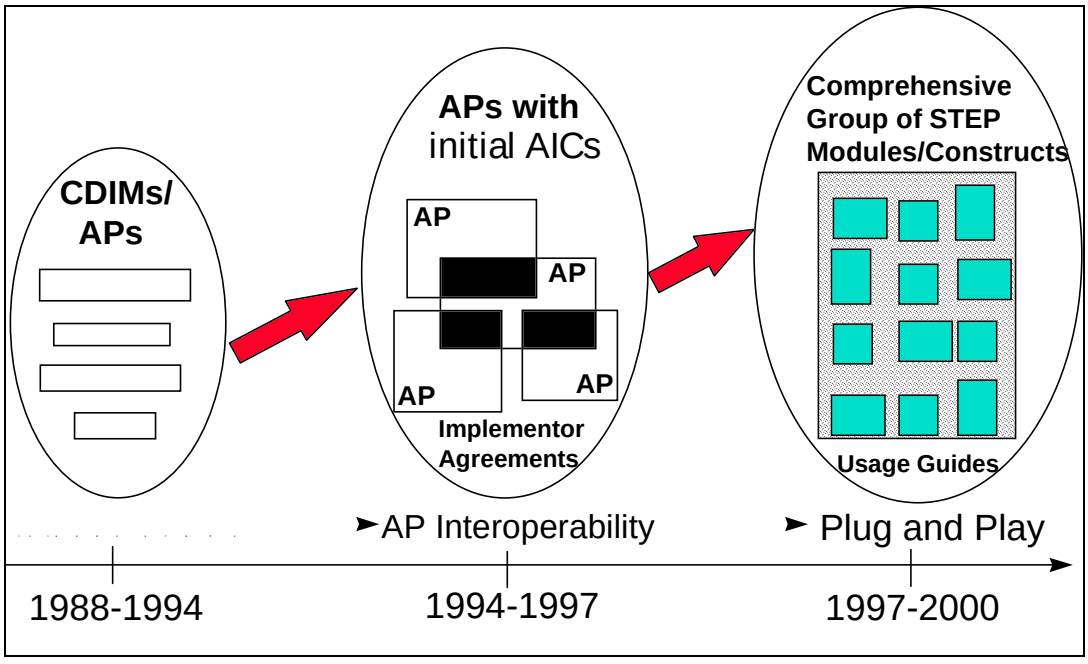


Figure 47: The move towards Modules

[1] An exception to this might be those domains where purely a parameterised description is needed

WG10 within the SC4 group of STEP have generated a number of ideas on this front. Most of this work has been carried out by the PDES Inc. group and been reviewed by the various STEP centres around the world. These are presented below.

The driving force behind this has been pointed out already. The concept of an Application Module is thus being used within STEP to modularize the standard into reusable parts. These modules will then be used within Application Protocols to define the exchange standards for a particular application context. The documentation of a Module is somewhat similar to that of an AP, thus the phrase "mini-AP" has been coined.

4.3.4 Types of Module

Application Modules will exist in a number of layers from very generic modules to very application context specific modules. Each module will define an information model for one or more concepts. Where they require concepts already defined in other modules they will 'use' or link to those other modules. For instance an Approvals module may need a Date Time module.

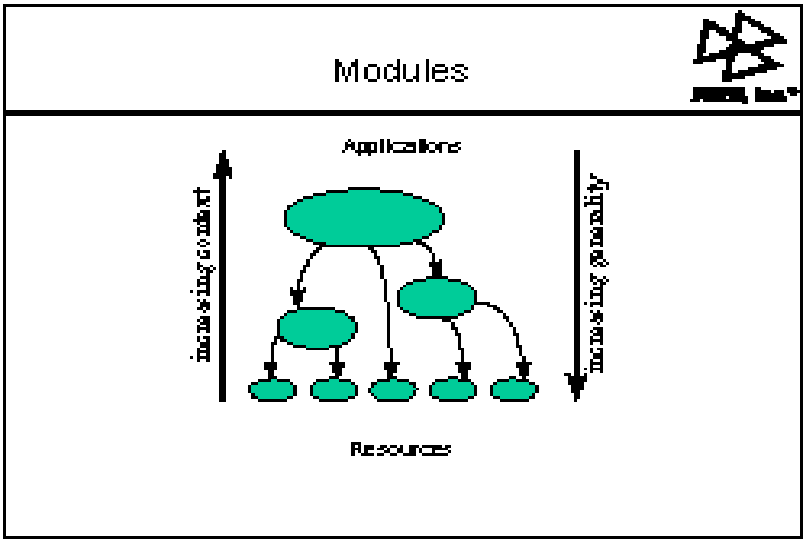


Figure 48: Module Layering

4.3.5 Module and AP Structuring

The structure of the documentation of a AP which uses a module has been proposed as shown in **Figure 49: AP Documentation**. Note that the chief differences here between current AP documentation and that proposed is that Clause 4 identifies the modules required from the Industry Requirements and that Clause 5 only contains the AIM shortform. The new structure allows existing modules to be reused whilst allowing industry specific requirements to be identified and the appropriate modules to represent them to be specified. Only the AIM shortform is required since each module has it's own mapping table.

Figure 50: AM Documentation shows the documentation of each module. Obviously, the scope is much narrower at the lowest levels of AM, and Clause 4 provides a mechanism to not only document the application objects and associated data, but also allows to formally show which other AM's are used to complete the module being defined (layering). Lastly the

application objects are mapped to the resources through the mapping table in Clause 5 similarly to that done today but in the APs.

Application Protocol - Document Content

Forward

Introduction

1 Scope

2 Normative References

3 Definitions and Abbreviations

4 Information Requirements from Modules

4.1 Units of Functionality with associated modules

4.2 Industry Specific Information Requirements

4.2.1 Requirements Definition

4.2.2 Mapping of Industry Specific Requirements onto Requirements in Modules

4.3 AP Application Assertions

5 Application Interpreted Model

5.1 AIM EXPRESS Short Form Listing

Annexes

A Implementation Method Specific Requirements

B Information Object Registration

C Application Activity Model Optional

C.1 Application Activity Model Definitions and Abbreviations

C.2 Application Activity Model Diagrams

D Application Reference Model (Req. for a specific AP ARM)

E AIM EXPRESS listing SF and all necessary schemas

F App Protocol Implementation and Usage Guide

G Technical Discussions

H Bibliography

Figure 49: AP Documentation

Application Module - Document Contents

Foreword

Introduction

1 Scope

2 Normative references

3 Definitions and abbreviations

4 Information requirements

4.1 Units of functionality

4.2 Referenced AM ARMs

4.3 ARM type definitions

4.4 ARM entity definitions

4.5 ARM rule definitions

4.6 ARM function definitions

5 Module interpreted model

5.1 Mapping table

5.2 MIM EXPRESS short listing

Annexes

A AM MIM short names

B Information object registration

C ARM EXPRESS-G

D MIM EXPRESS-G

E AM ARM and MIM EXPRESS listings

F Application module implementation and usage guide

G Technical discussions

H Bibliography

Figure 50: AM Documentation

4.3.6 Module Development

The following diagram shows how AP203 is being modularized to allow it to extend its scope to include additional functionality such as colors, layering information, dimensions and tolerances.

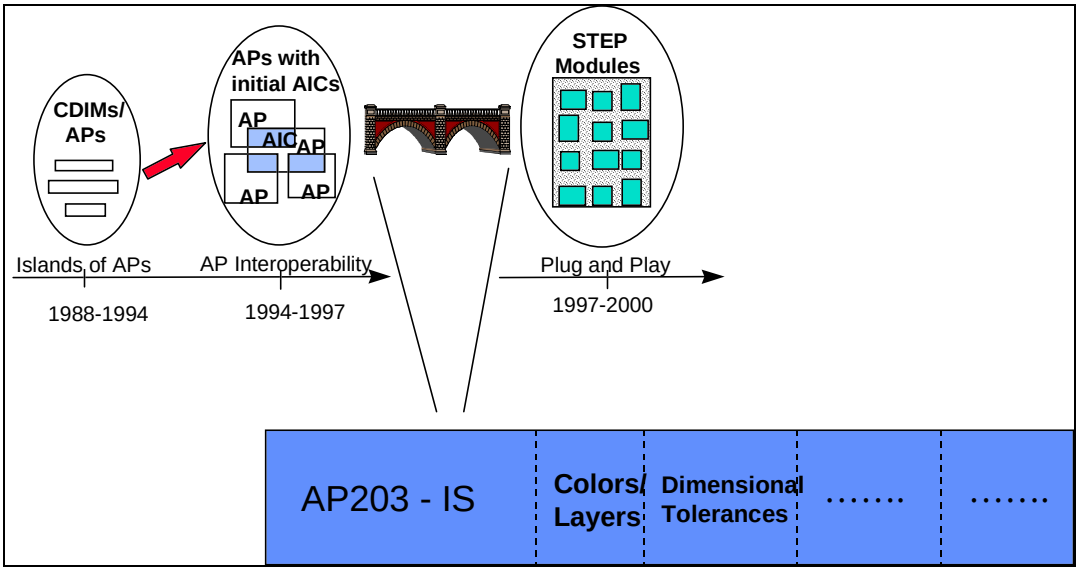


Figure 51: Modularization of AP203

4.3.7 Targets for AP Modules

These are the current perceived goals for the Modules being developed.

- A module should be 100 or less pages (not a hard rule)
- A module should be a single unit of functionality
- A module is basically an AP-LET
- It has all the components of an AP
- This is similar to the ProSTEP proposal for AICs which would have added ARMs
- It is different from an AIC because:
    - an AM contains an ARM
    - can be created as an interpretable unit even if no new rules are added
    - does not require at least 2 APs to exist
- It may be independent or dependent
- A module may require another module for legal implementation
- No one should implement colours, layers and groups by itself
- This could be legislated by an Application Protocol
- Modules allow vendors to write code once and use it many times

4.3.8 Major Differences between AICs and AMs

There are a number of differences that can be highlighted between the Module approach being developed with that of AICs. These are shown below in the following two figures (Figure 52: AIC Approach and Figure 47: The move towards Modules).

Figure 52: AIC Approach

- 2 or more APs required
- Concepts re-documented
- Use of constructs may differ
- No ARM
- No mapping to IRs
- Not testable
- Not implementable
- Not complete
- Root node required
- All rules local to root
- Common interpretation not sufficient
- AP uses entire AIC

Figure 53: AM Approach

- A single, or no, AP required
- Concept documented only once
- A single use
- ARM documents use
- Mapping included
- Test suite based on AM
- Potentially implementable
- Complete
- No artificial root nodes
- Global rules allowed
- Common interpretation is sufficient
- AP uses entire AM

4.3.9 Technical Considerations

Below we highlight the major technical differences between AM's and AP's as presently understood;

- May subtype multiple IR entities — AM : YES AP : NO
- May add cardinality global rules — AM : YES AP : YES
- May add subtype and constraint — AM : YES AP : NO
- May add subtype and derive — AM : YES AP : NO
- May complete mgmt resources — AM : YES AP : YES?
- May add value global rules — AM : YES AP : YES
- May add subtype and redeclare

4.3.10 Ongoing Discussions

Some of the ongoing discussions with respect to Modularization include;

- how to overcome the SELECT type problem?

  The problem here is that a SELECT type might well introduce entities which reside in different modules. However, when choosing a module for a specific purpose or requirement, it has been found that not all of the SELECT type contents are needed. Therefore, the result is that because of using one particular module, many

other modules are “pulled in” when these may not be needed. This in turn makes implementation more lengthy and complex since many of the entities would never be used.

- should contain both the ARM and AIM information within the Module?

  It has been argued that because the ARM is not implemented that it is not needed to be part of the formal definition of the standard AM. However, much of the intention of the model is lost in the AIM since most terminology from the ARM is replaced during the interpretation process. It is also the case that much effort goes into producing the ARM and that such information should be retained. It might also help to have information present when implementing the software to support the requirements.

- how should Application Modules be introduced into the ISO standardisation process?

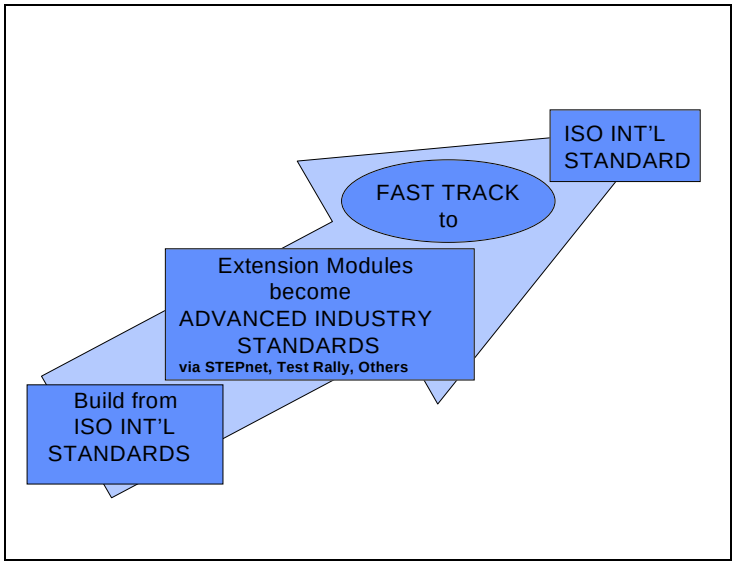


Figure 54: Module Standardisation Process

At present the proposals for AM development and guidelines for use have been presented as white papers by (primarily PDES Inc.) via the ISO TC184/SC4/WG10 group. The discussion has largely been resolved in that the Modular Approach will not replace the traditional approach of AP development. Moreover, it is being presented as an alternative. However, should this fail to gain adequate support the fall-back position would be to launch a separate standard in parallel to STEP, much like Parts Library (PLIB) or Oil and Gas (previously POSC/CAESAR initiative)..Currently this has resulted in a New Work Item being submitted by PDES Inc. through it’s role as an Active Liaison to SC4. This may have some advantages over the normal ISO standardisation process if presented as an industry standard.

- how should Application Module information be disseminated?

  Along side these white papers, there has been a major dissemination drive at each STEP meeting resulting in broad acceptance of the approach. Most still see this as something that will be used by future APs and will therefore not affect them.

5 Annex

5.1 ABBREVIATIONS

For the purposes of this Part, the following abbreviations apply:

AAM	Application Activity Model
AIM	Application Interpreted Model
AP	Application Protocol
ARM	Application Reference Model
BB	Building Block
CAD	Computer Aided Design
CAM	Computer Aided Manufacture
EMSA	European Marine STEP Association
IMO	International Maritime Organisation
PICS	Protocol Implementation Conformance Statement
SCM	Ship Common Model
SI	Système International
SOLAS	Safety of Life at Sea
SPM	Ship Product Model
UoF	Units of Functionality

5.2 BIBLIOGRAPHY

[N327] Koch Th., de Bruijn W.: ISO TC183/SC4/WG3 document N327 – Modelling Framework for Shipbuilding.

[AP Guidelines] Guidelines for the development and approval of STEP Application Protocols; version 1.2; ISO TC 184/SC4/WG4/ N506, May 5, 1995.

[LR12ADP] Turner, J.: STEP AEC Shipbuilding, AP Cross Reference List, October 17th 1997

[Common AAM] Kendall, J: ISO TC 184/SC4/WG3 N511 - Shipbuilding Common Activity Model, dated 14 January 1996.

[SCM-97] Turner, T: Ship Common Model, dated Version dated 30 July, 1997.

[Guidelines-96] Haenisch,J: ISO TC184/SC4/WG3/N498 - AP Development Guidelines for Shipbuilding, dated 17/07/96

[Giacometti 90a] Giacometti F., Chang T-C., A model for parts, assemblies and tolerances, pre-prints of the first IFIP W.G.5.2 workshop on design for manufacturing, Enschede, 1990.

[Giacometti 90b] Giacometti F., Chang T-C., Object-oriented design for modelling parts, assemblies and tolerances, proceedings Technology of Object-Oriented Languages and Systems (TOOLS), Paris, 1990, 243-255.

[Guarino 1997] Guarino N., Some organising principles for a unified top-level ontology, LADSEB-CNR Int. Rep 02/97

[Kiriyama 91] Kiriyama T., Tomiyama T., Yoshikawa H., The use of qualitative

physics for integrated design object modelling, ASME Conference on Design Theory

and Methodology, DE-Vol. 31, 1991, 53-60.

[Salomons, 1995] Salomons O.W., Computer support in the design of mechanical products, Ph.D. thesis, University Twente, January 1995

[Shah 90a] Shah J.J., An assessment of features technology, CAM-I report P-90-PM-02, 1990.

[Shah 90b] Shah J.J., Philosophical development of form feature concept, CAM-I

report P-90-PM-02 , 1990, 55-70.

[Shah 91] Shah J.J., Conceptual development of form features and feature

modellers, Research in Engineering Design., Vol.2, 1991, 93-108.

[Sodhi 91] Sodhi R., Turner J.U., Representing tolerance and assembly information

in a feature based design environment, Advances in Design Automation, DE Vol.

32-1, ASME, 1991, 101-106.

[Wingerd 91] Wingerd L., Introducing form features in product models, a

step towards CAD/CAM with engineering terminology, Licentiate Thesis, Dept.

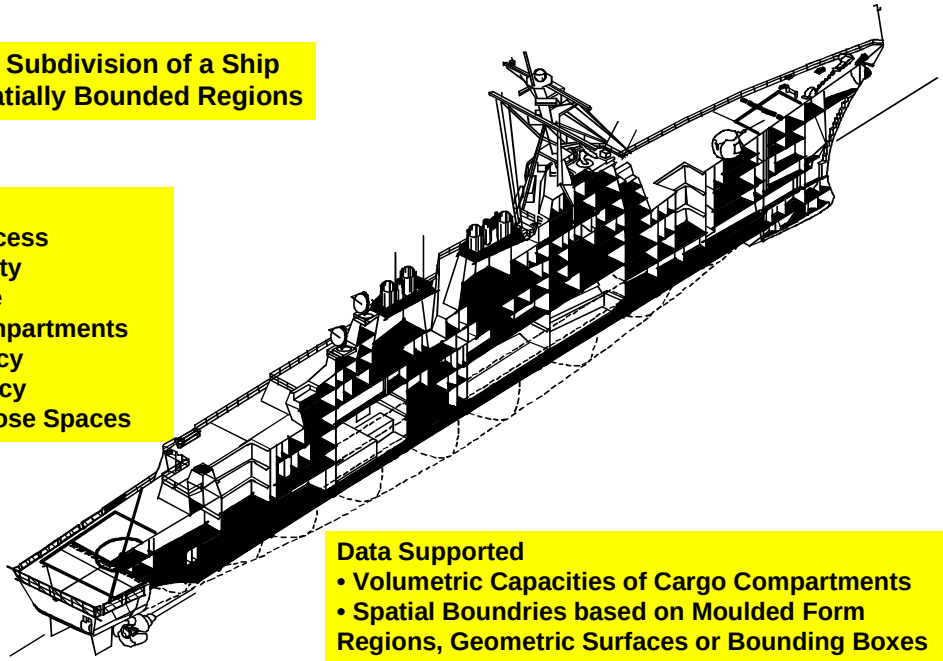
of Manufacturing Systems, Royal Institute of Technology, Stockholm, 1991.

5.3 SHIPBUILDING APS UNDER DEVELOPMENT

AP 215: Ship Arrangement

General Subdivision of a Ship into Spatially Bounded Regions

Zone Boundries
- **Controlling Access**
- **Design Authority**
- **Cargo Stowage**
- **Machinery Compartments**
- **Crew Occupancy**
- **Space Adjacency**
- **Common Purpose Spaces**



Data Supported
- **Volumetric Capacities of Cargo Compartments**
- **Spatial Boundries based on Moulded Form Regions, Geometric Surfaces or Bounding Boxes**
- **Calculation of Effect on Structural Systems based on Magnitude and Location of Cargo Loads**

AP 216: Ship Moulded Forms

Surface, wireframe and offset point representations

Design, Production and Operations lifecycles

General characteristics

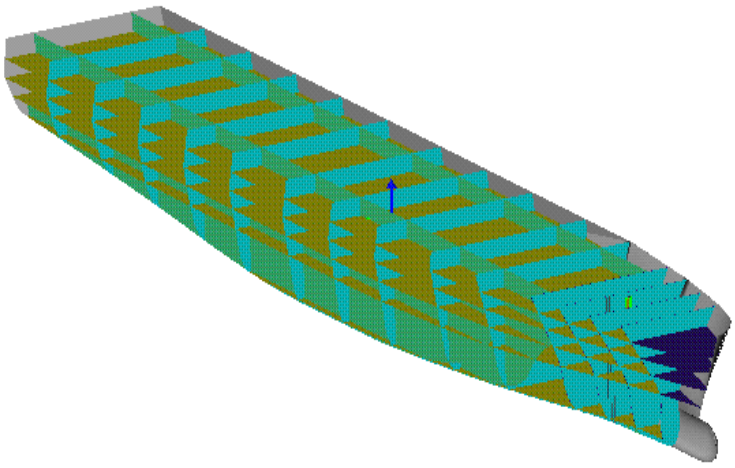
Main dimensions

Hullform geometry

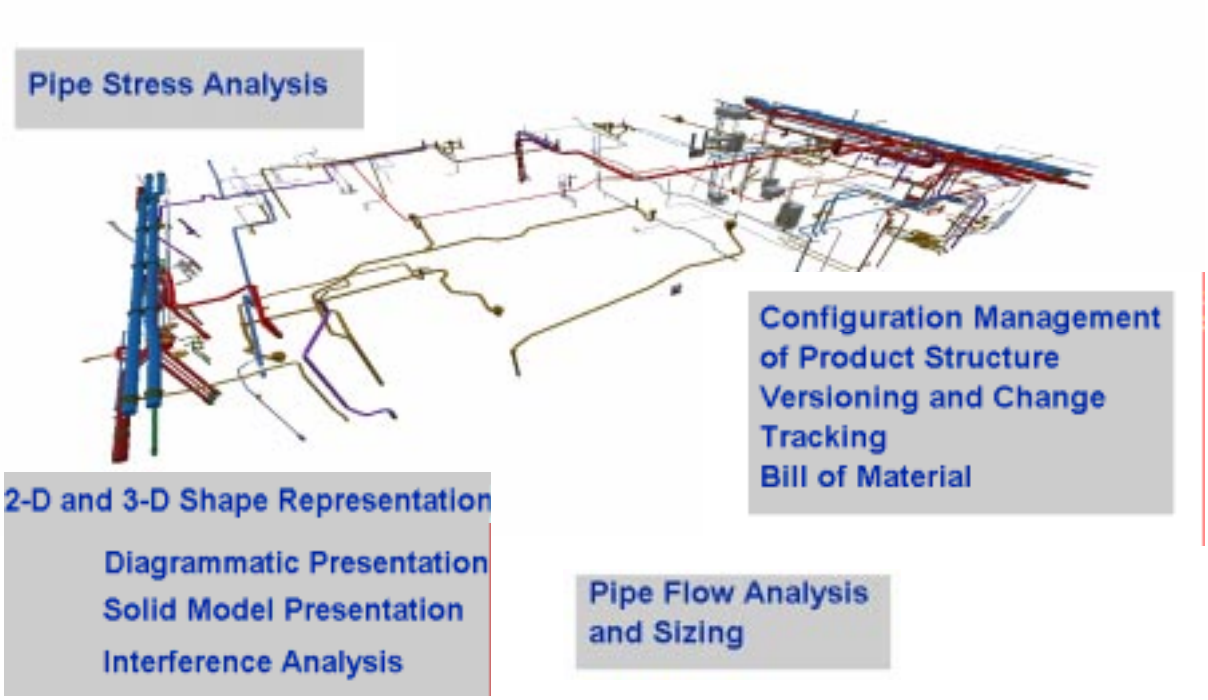
Major internal surfaces

Hydrostatics

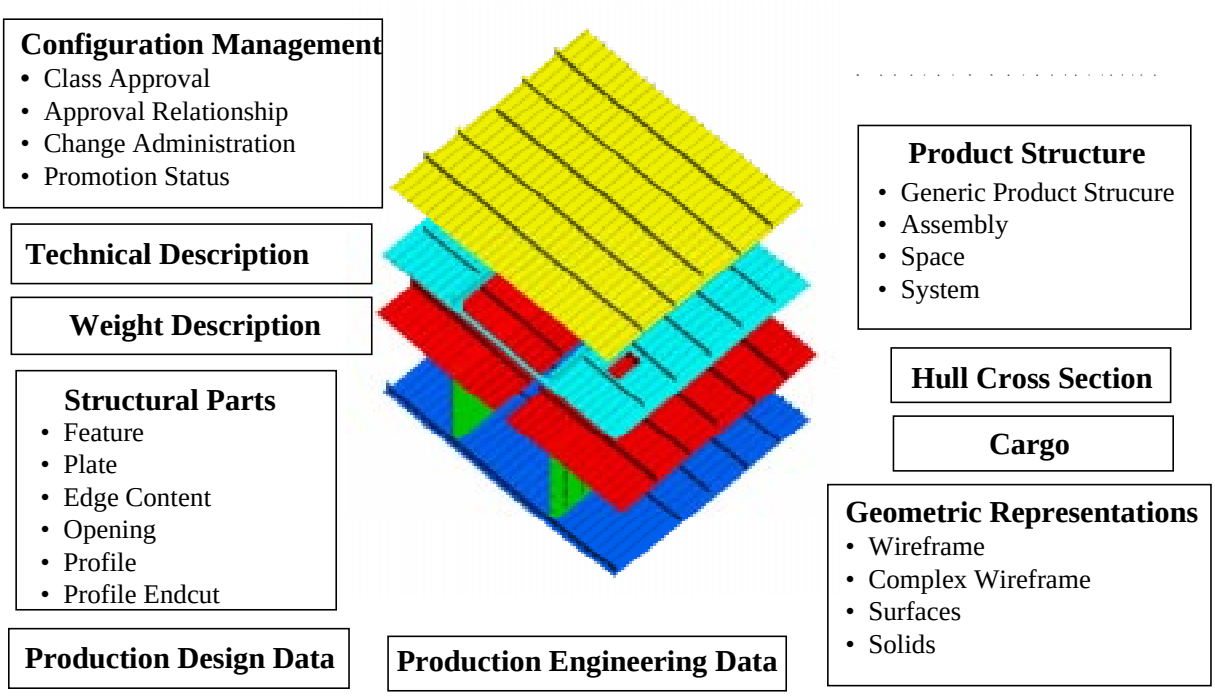
Intact Stability tables



AP217: Ship Piping

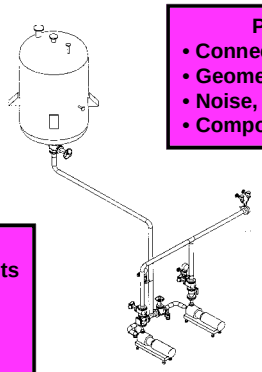


AP 218: Ship Structures



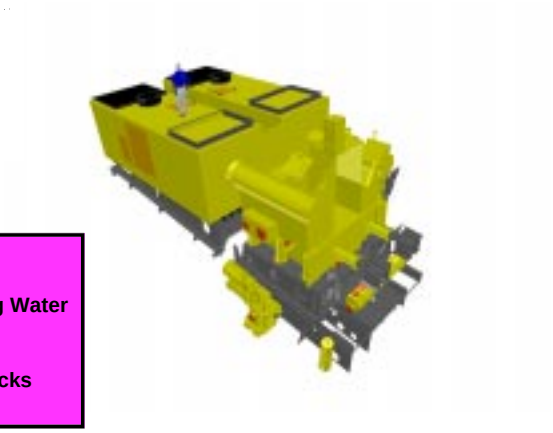
AP 226: Ship Mechanical Systems

System Components
- Air Supply to Engine Room
- Exhaust Gas Removal
- Fuel Oil Treatment and Supply
- Engine Lubricating and Cooling
- Propulsion - Main Shaft/Couplings
- Maneuverability - Rudder/Thrusters
- Electrical Generation



Product Definition Information
- Connectivity Between Components and Systems
- Geometry, Materials, Topology and Tolerances
- Noise, Vibration and Shock Characteristics
- Component/System Life Cycle and Operational History

Other Data Supported
- Functional Description of Components
- Quality Assurance Information
  - Availability
  - Reliability
  - Maintainability
- In-Service Inspection and Maintenance



Mechanical Components
- Main and Auxiliary Engines
- Pumps - Fuel Oil/Lubricating Oil/Cooling Water
- Air Compressors and Receivers
- Heat Exchangers
- Deck Machinery - Winches/Cranes/Derricks
-

5.4 COMMON APPLICATION ACTIVITY MODEL

This appendix presents the Shipbuilding Common Application Activity Model (AAM), with the accompanying definitions of the terms describing the information exchanged and the titles of the activities performed. Also included is the node tree that gives an overview of all the pages in the AAM and their hierarchy.

The Common AAM are the higher levels which are common to all the Shipbuilding APs. Each of the APs will contain the common AAM and expand the lower levels where necessary.

The AAM is a graphical representation of the activities carried out in a process and the information flows required between them. The AAM is used to aid identification of these flows of information and therefore clarify the scope and information requirements of the AP. The modelling of this AAM is done using the IDEF-0 (ICAM Definition Language 0) notation.

The application activity model (AAM) in this Annex is intended to provide a common basis from which all of the Shipbuilding Application Protocols can be built and extended according to the individual domain requirements. Such AAMs aid in understanding the scope and information requirements defined for each Application Protocol. This AAM provides the context of all the Application Protocols and as such covers activities which go beyond the scope of them individually. The model is presented as a set of definitions of the activities and the data, and a set of activity figures.

The viewpoint of the application activity model is of an observer of the global ship development process. This activity model identifies the life cycle activities across all shipbuilding APs with extensions and emphasis detailed in each AP where appropriate. Activities relevant to the shipbuilding lifecycle that are do not fit with this activity model but are required and are detailed in other shipbuilding application protocols, should be brought to the attention of the editors of this document so that this generic model can be amended where relevant.

The following definitions describe the activities, inputs, outputs, controls and modifiers which interact as shown in the forthcoming diagrams.

A.1.1 approved design : The approved design is the final design to be submitted as an offer.

A.1.2 arrangements : The arrangements of the ship are the ship’s compartments and spaces. Any description of arrangements will include associated definitions of purpose for the compartment or space.

A.1.3 assemble ship : the activity that assembles the modular units, the serviced parts and additional material that result from the production of steel sub-section. The result is an assembled ship, that still has to be tested.

A.1.4 availability, reliability and maintainability information : The information about the components that is required to install them in the ship and is required for planned maintenance.

A.1.5 basic hull parameters : Estimated principal dimensions based on historical data or preliminary design development.

A.1.6 budget : the cost constraint on the design building and maintenance of the ship.

A.1.7 calculate cost of ship : This activity describes creation of negotiating documents based on technical product data and their estimated manufacturing cost. The results of this activity may contain sale price documents, financing support plan and documents describing funding and possible loans.

A.1.8 certificates : The certificates issued by the Classification Society on completing the ship.

A.1.9 check design against rules and regulations : This is the top level activity for the approval of the primary design as part of the approval and certification process. The content of this activity is the same for all ships when it comes to conformance with Main Class Rules, but varies when it comes to additional class rules (type of vessel) and register notations. The activities performed are tailored to the rule requirements for general arrangement and global strength. This part of the approval is necessary before the yard can start ordering steel.

A.1.10 Classification Society : An organisation that enhances the safety of life and property at sea by providing rules, regulations and personnel for assessing and classifying ships during their lifecycle.

A.1.11 complete and approve design of machinery : The selection, arrangement and approval of the power plant in terms of the main engine, associated propulsion system and its auxiliary machinery.

A.1.12 complete and approve design of outfitting and distribution systems : The selection and approval of the necessary outfitting equipment. The selection is based mainly on former designs and in accordance with the requirements. It also contains the layout of the different types of distribution systems such as piping and HVAC.

A.1.13 complete and approve design of ship structure : The completion and approval of the ship structural design.

A.1.14 complete and approve ship design : The production and approval of ship design product data, documents and the classification drawings using the preliminary design from the bid preparation, as well as the required rules and regulations. The result of this activity is the approved design and the production and delivery schedule.

A.1.15 consultants : Organisations that provide specific services to shipyards, ship owners and classification societies during the ship lifecycle.

A.1.16 contract : The contract is the output from the activity which involves placing the order for the ship. The contract is used as a constraint in subsequent activities such as final design and approval and production.

A.1.17 cost : The calculated cost of the ship based on the cost of material and labour.

A.1.18 create preliminary design : All design activities relevant in a very preliminary stage of ship design in consideration of classification rules, national/international demands, shipyard constraints and owner requirements. The aim of this task is to make a shipyard offer.

A.1.19 create preliminary general arrangements : The activity that produces the preliminary compartmentation plans from the preliminary hull form definition.

A.1.20 create preliminary hull form : The activity that is the first step of designing a ship. Using parent ships main dimensions and form parameters one or more preliminary hull forms will be generated.

A.1.21 create preliminary machinery design : The activity that produces the preliminary designs for the ship machinery; including the prime mover, shaft system, fuel system, power systems and cargo handling equipment.

A.1.22 create preliminary outfitting design : The activity that produces the preliminary design for the ship's outfitting, including distributed systems, such as piping and electrical systems.

A.1.23 create preliminary structure design : The activity that produces the preliminary steel structure design, including the arrangement of the primary structural members.

A.1.24 decide post-sales & maintenance support : The activity that puts together the maintenance package for the ship. This is part of the tender document and includes the post sales support.

A.1.25 decommission and disassemble : All activities relating to the last stage of the ship's lifecycle. It consists of the decommissioning and dismantling of the ship.

A.1.26 design schedule : Data that controls the time from the design phase to production.

A.1.27 distribution and outfitting design : The design of the distribution systems (electrical and piping) and the outfitting.

A.1.28 estimate hydrodynamics and powering : The activity that approximates hydrodynamic properties data calculations such as resistance, propulsion, seakeeping and manoeuvrability for the preliminary hull form.

A.1.29 evaluate request & schedule bid : This describes the activities of the shipyard when evaluating the inquiry of the ship owner for a new ship.

A.1.30 feedback : The outputs from activities which then feed back and modify previous activities in the lifecycle on the current or subsequent ships.

A.1.31 finalise and approve general arrangements : The activity that details the general arrangement after having created a draft layout. The ship's systems are described by a compartment and access drawing showing the location, the access, and the size of the different compartments.

A.1.32 finalise and approve hull form : The activity in which the hull form is finalised from the preliminary design. The result is a final and approved hull form design.

A.1.33 finalise and approve hydrodynamics and powering : This includes all relevant hydrodynamic calculations such as resistance, propulsion, seakeeping and manoeuvrability.

A.1.34 general arrangements : The space arrangement plan from the preliminary design stage.

A.1.35 historical data from previous designs : Data held by the shipyard or model basin on previous ship designs and used to estimate the hydrodynamics, powering requirements and sea-keeping.

A.1.36 hull form sections : The design of the hull moulded form at planar sections taken along the longitudinal axis of the ship.

A.1.37 hull moulded form : The definition of the shape of the hull of the ship, resulting from the addition of the aft-body, mid-body and fore-body definitions, which does not take into account the thickness of the material from which the hull is made.

A.1.38 hydrodynamics & powering results : The results of calculations and model basin tests. They contain resistance, propulsion, propeller performance, brake power, service speed, sea keeping and manoeuvrability data.

A.1.39 knowledge and experience : The previous experience and knowledge of companies involved throughout the ship lifecycle.

A.1.40 laws, rules and regulations : National laws, statutory regulations and classification society rules that are used to control the design, manufacture, operation, maintenance and scrapping of the ship.

A.1.41 list of required certificates : The result of placing an order, this is the list supplied by the owner for certificate requirements.

A.1.42 loading and stability manual : a booklet which is placed on board the ship for the information of the master, which enables him or her to load the ship within prescribed limits, relating to strength and stability.

A.1.43 machinery design : The design drawings and electronic models of the ship mechanical systems. An output from the final design process.

A.1.44 machinery weights : These outputs are the results of several calculation and design activities which result in an estimated weight for all machinery.

A.1.45 manufacturing restrictions : A constraint on the ship construction and design processes governed by available technology and shipyard facilities.

A.1.46 material list : The list of raw materials needed to manufacture the ship. A result of the final design process.

A.1.47 modifications from machinery : Modifications to the hydrodynamics and powering due to feedback from the preliminary machinery design.

A.1.48 modifications to hull form : Modifications to the hull shape due to feedback from hydrodynamics and powering results and the final design process.

A.1.49 modular units : sub-sections of the ship complete with machinery and outfitting which will be assembled to create the final product.

A.1.50 offer : The result of the preliminary design process. It will contain the shipyard's data for producing the requested ship.

A.1.51 offer guidelines : The offer guidelines include the data necessary to make an unconditional offer to the ship owner

A.1.52 operate and maintain a ship : The activity that describes the running and maintenance of the ship during its service lifetime.

A.1.53 operational information : Accumulated information during the operation phase of the ship used for maintenance and in the final scrapping stage.

A.1.54 owner : The organisation which requests, orders, takes delivery of and, for the purposes of this model, operates the ship.

A.1.55 owner request, requirements : The requirements document that is submitted to the shipyard by the owner upon the invitation to tender.

A.1.56 perform ship lifecycle : All of the lifecycle activities associated with a ship.

A.1.57 place order : The owner places an order for a ship from the bids that have been submitted. From this a contract is awarded.

A.1.58 planned maintenance system : Data created during the final design process and used during the operation and maintenance of the ship.

A.1.59 pre layout : The very initial layout of the ship which is produced during the bid evaluation stage and is the basis for the preliminary design.

A.1.60 preliminary design : The preliminary design is that which is completed in the phases leading up to the submission of the tender.

A.1.61 preliminary general arrangements : The definition of the ship general arrangements as a result of the preliminary design process.

A.1.62 preliminary hull form : The definition of the hull form, as a result of the preliminary design process. Used in the offer documents and for preliminary compartment design, hydrodynamics and powering calculations.

A.1.63 preliminary machinery design : The definition of the ship mechanical systems. Used early to estimate the noise, speed and vibration and to estimate the machinery weights.

A.1.64 preliminary machinery, structure and outfitting design : Feedback consisting of the preliminary designs for machinery, structure and outfitting and furnishing. This allows the creation of preliminary general arrangements.

A.1.65 preliminary outfitting design : The definition of the ship’s outfitting and accommodation, resulting from the preliminary design process.

A.1.66 preliminary structure design : The definition of the preliminary ship structure during the preliminary design process.

A.1.67 prepare bid : This activity includes all activities of the yard regarding preparation and submission of the offer to the ship owner for the ship to be built.

A.1.68 present offer : The activity concerned with presentation of the offer to build the ship to the prospective ship owner.

A.1.69 produce and approve reference documents : the technical documentation for the ship is produced using production information. The output includes the loading and stability manual which is approved by the Classification Society.

A.1.70 produce and inspect a ship : This activity includes high-level activities such as produce, monitor and inspect ship production. Inspect, means the controlling of all activities throughout the whole production life cycle of a ship.

A.1.71 produce modular build units : this activity covers the production of the modular units which will make up the completed ship. They are produced from the steel-subsections and their production is controlled by the schedule, contract, the approved design, and any manufacturing restrictions. The results of the activity are the modular units which are assembled into the ship.

A.1.72 produce steel sub-sections : this activity covers the production of the steel sub-sections which make up the structure of the completed ship. This is controlled by the schedule, contract, the approved design, and any manufacturing restrictions.

A.1.73 product component information : The technical data about the components that will be incorporated into the ship. These are taken into consideration when the preliminary designs are being made.

A.1.74 production and delivery schedule : The schedule according to which the ship is manufactured and delivered.

A.1.75 production information : information describing a product, e.g. dimensions, mechanical properties, workshop information.

A.1.76 propeller design : The design of the propeller or propulsor as a result of the hydrodynamics and powering calculations. The design controls some of the machinery design activity.

A.1.77 quality assurance : the rules applied by an organisation within the shipyard that has the task to audit the shipyard organisation and applied processes in a manner such that the quality of the resulting product is assured.

A.1.78 request a ship : The first activities of a ship owner when intending to order a ship. Having definite ideas regarding appearance and functionality of the ship, the owner expresses these ideas in an inquiry to the shipyard.

A.1.79 request for production changes : Changes that are requested to the ship design as a result of production experience or difficulties with the realisation of the ship design.

A.1.80 resistance and shaft power : The result of the activity to estimate hydrodynamics and powering. Resistance and shaft power is a constraint on the creation of the preliminary hull form.

A.1.81 resources : The shipyard, classification society, and outside consultants.

A.1.82 schedule : The schedule is formed as a part of the final design process. It governs the timing of the production phases.

A.1.83 scrapping plan : The document used to schedule the time and resources required to dismantle the ship.

A.1.84 ship : a large waterborne vessel whose design, manufacture and lifecycle operation is governed by the principles of naval architecture and in accordance with international and classification society regulations.

A.1.85 ship product model data : The product data of the accumulated throughout its lifecycle. Because scrapping is part of the lifecycle the ship is not an output, only the documented information and knowledge about the ship survives.

A.1.86 ship weight modifications : Modifications to ship weight due to the preliminary structure design. This is fed back to modify the preliminary hull form and revise the preliminary general arrangements.

A.1.87 shipyard : An organisation that designs, builds, maintains, and repairs ships.

A.1.88 specify ship : All activities associated with the production of a detailed specification of the ship prior to a contract being placed.

A.1.89 steel sub-sections : the sub-sections of the steel structure which are outfitted with the machinery and distribution systems before assembly.

A.1.90 structural design : The design of the hull structure including hull, bulkheads, decks and stiffeners.

A.1.91 technical documentation : In case of maintenance the technical documentation of a system means part of the product description required to perform preventative maintenance, repair and failure analysis of that system. Technical information is an output which includes more detail information about material parts needed for producing the ship/system.

A.1.92 technical requirements : The owner's specifications that must be realised by the completed ship.

A.1.93 test results : maintenance test results are the results of functional tests carried out after the execution of maintenance actions.

A.1.94 test ship : this activity tests the actual ship against the design, contract and rules and regulations. The structure, is tested and sea trials are carried out. The test results are an output from this activity.

A.1.95 test structures : the steel structures are tested against rules and regulations and the design. The output is the test result documentation.

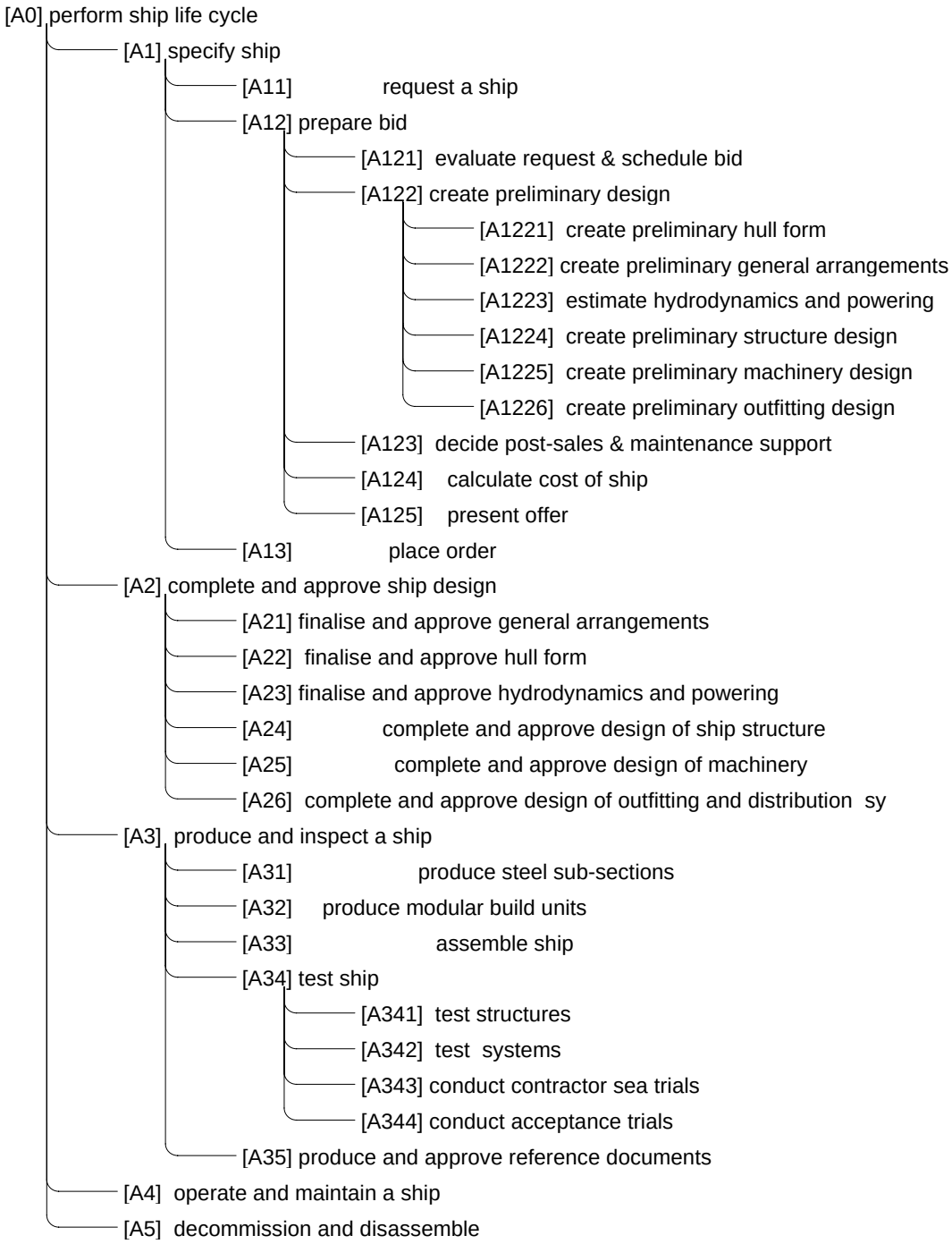
A.1.96 test systems : the ship's systems including outfitting, machinery and mission systems are tested against rules and regulations and the design. The output is the test result documentation.

A.1.97 transportation need : A constraint which determines the specification for the ship construction.

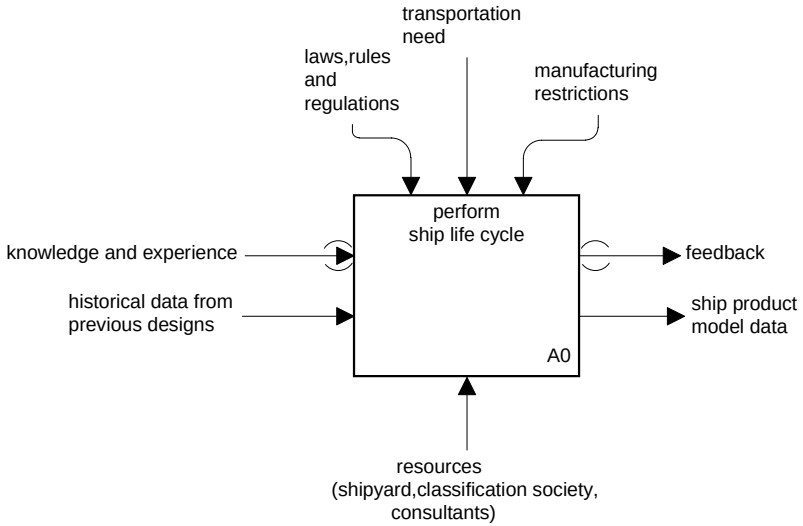
A.1.98 weights and centres of gravity : Weights and centres of gravity necessary for further calculations.

A.1.99 workload : The total effort required to build the chosen ship design as estimated by the shipyard and assisting consultants.

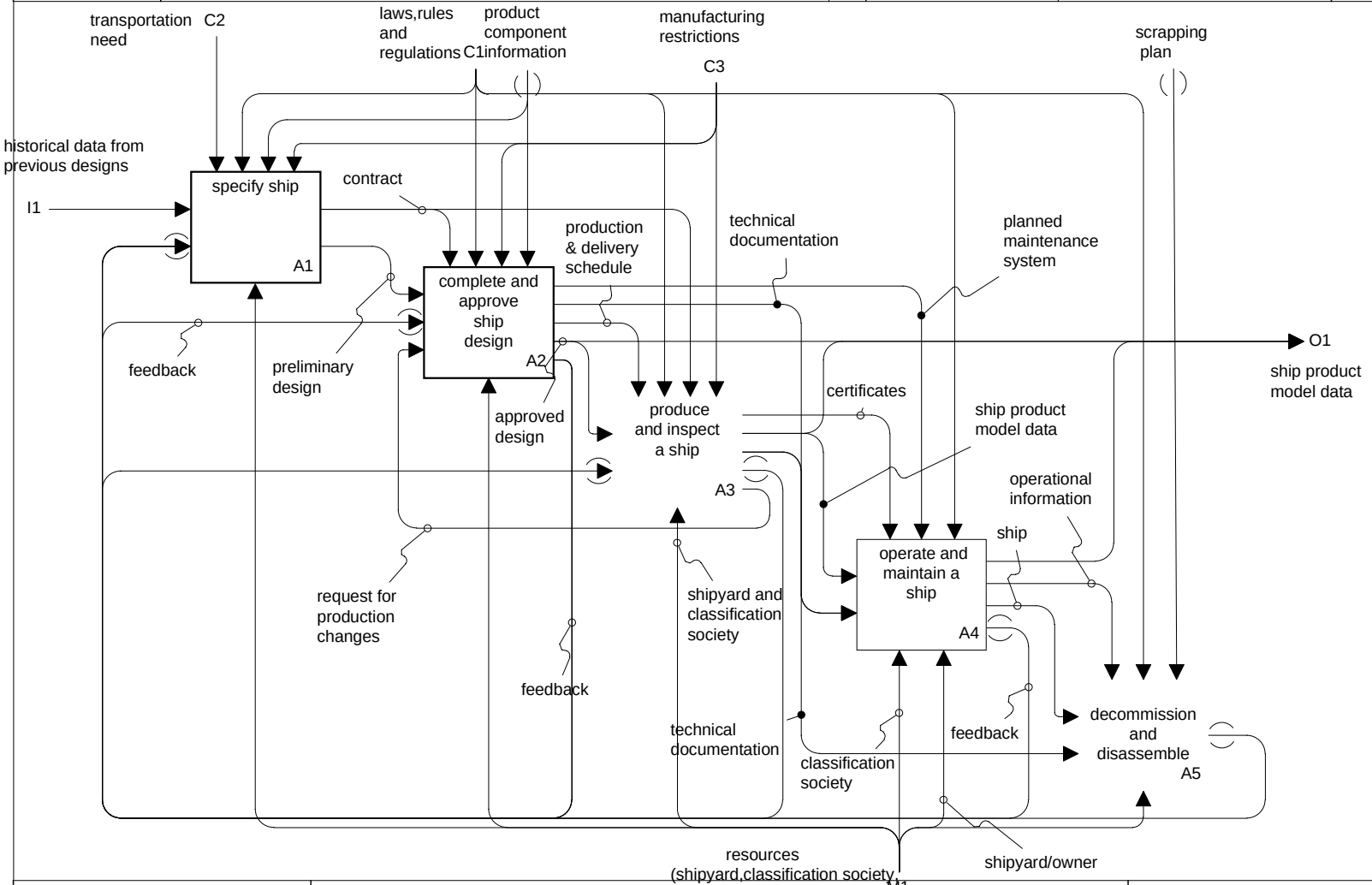
Activity Node Tree



USED AT:	EDITOR: John Kendall	DATE: 22/05/96	x	WORKING	READER DATE	CONTEXT:
	PROJECT: STEP Shipbuilding Common AAM	REV: 2.0		DRAFT		Top
	NOTES: 1 2 3 4 5 6 7 8 9 10			RECOMMENDED		
				PUBLICATION		

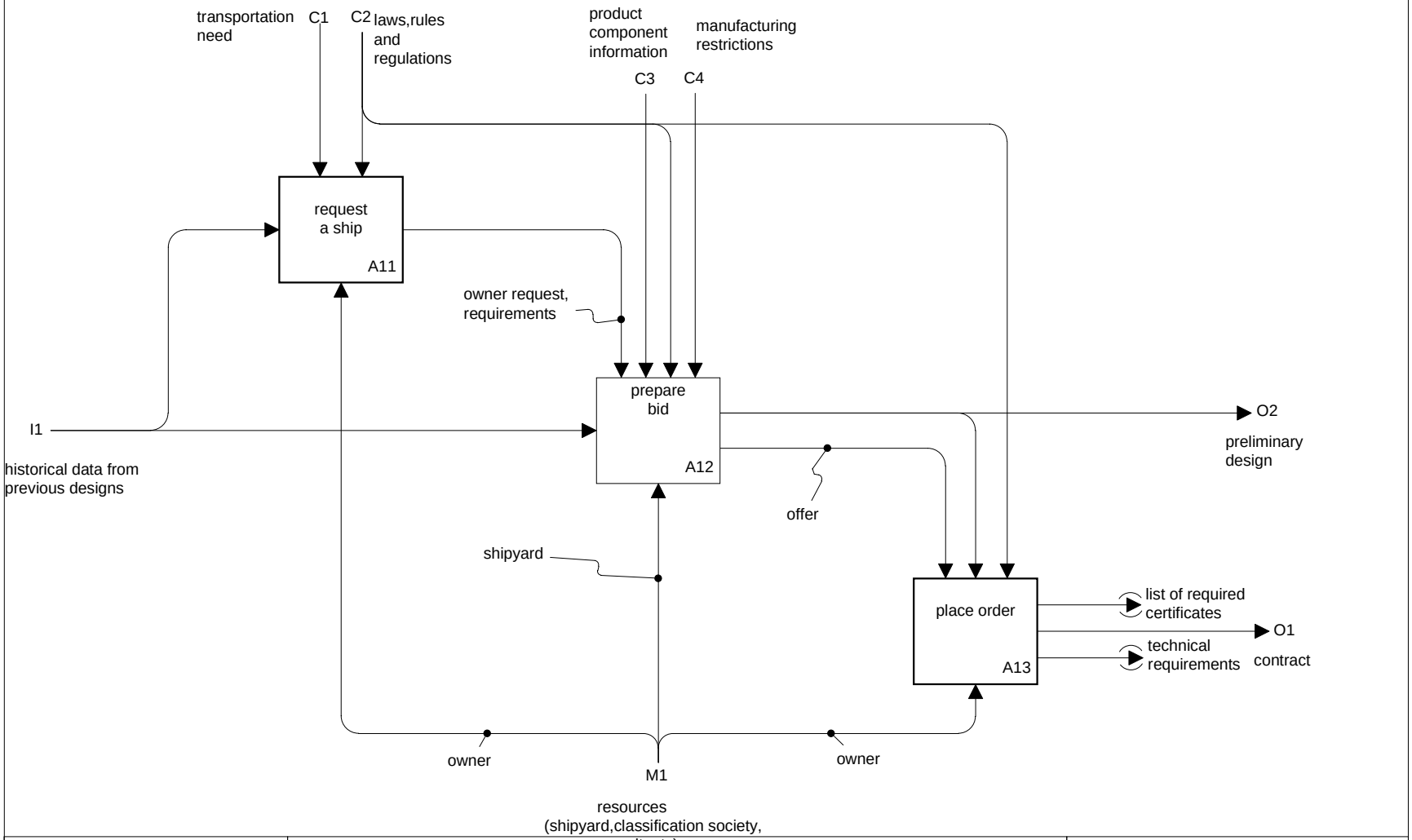


USED AT:	EDITOR: John Kendall	DATE: 22/05/96				CONTEXT:
	PROJECT: STEP Shipbuilding Common AAM	REV: 2.0	x	WORKING	READER DATE	
				DRAFT		
	NOTES: 1 2 3 4 5 6 7 8 9 10			RECOMMENDED		
				PUBLICATION		



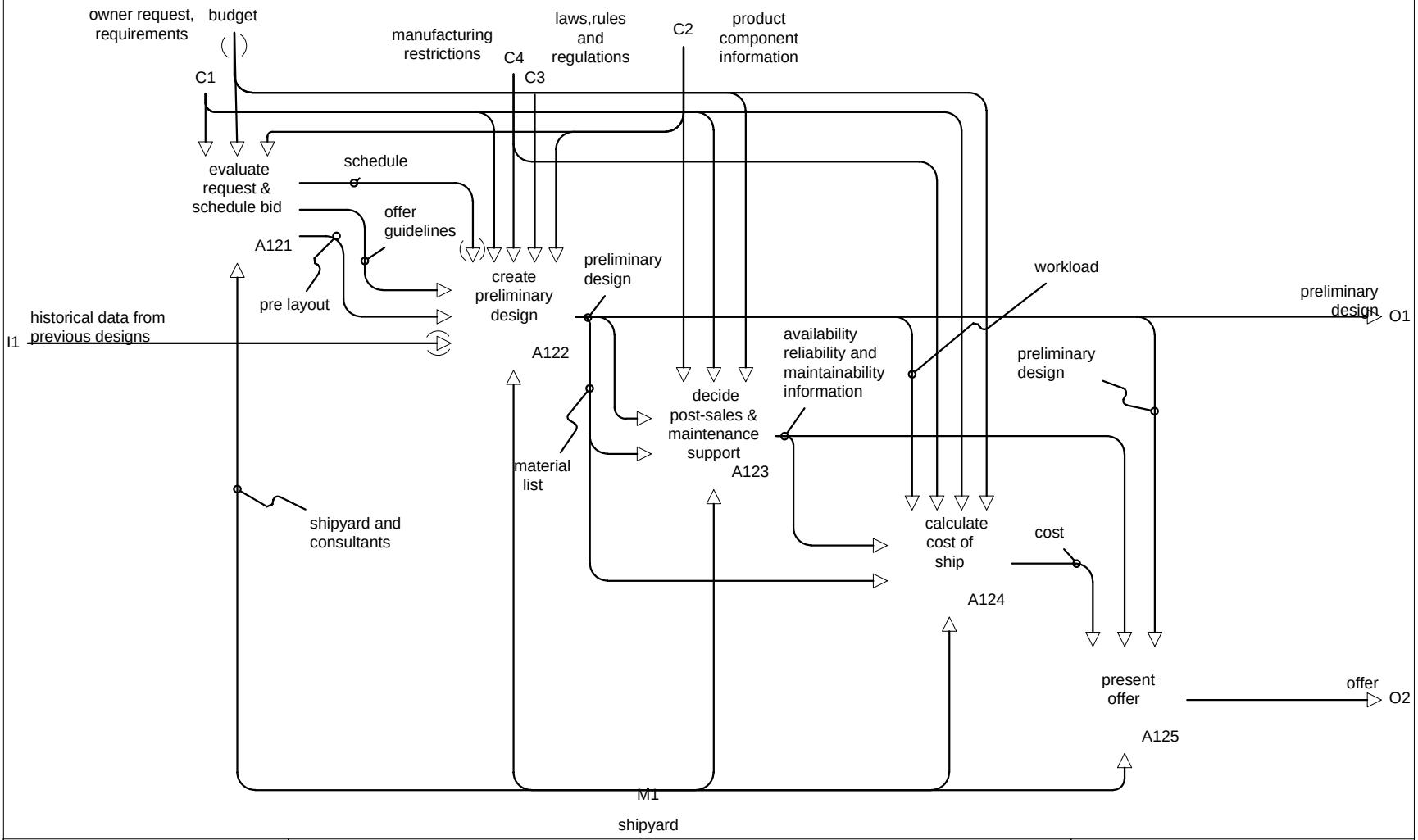
NODE: A0	TITLE: perform ship life cycle	NUMBER: P. 2

USED AT:	EDITOR: John Kendall	DATE: 22/05/96	x	WORKING	READER	DATE	CONTEXT:
	PROJECT: STEP Shipbuilding Common AAM	REV: 2.0		DRAFT			
	NOTES: 1 2 3 4 5 6 7 8 9 10			RECOMMENDED			
				PUBLICATION			



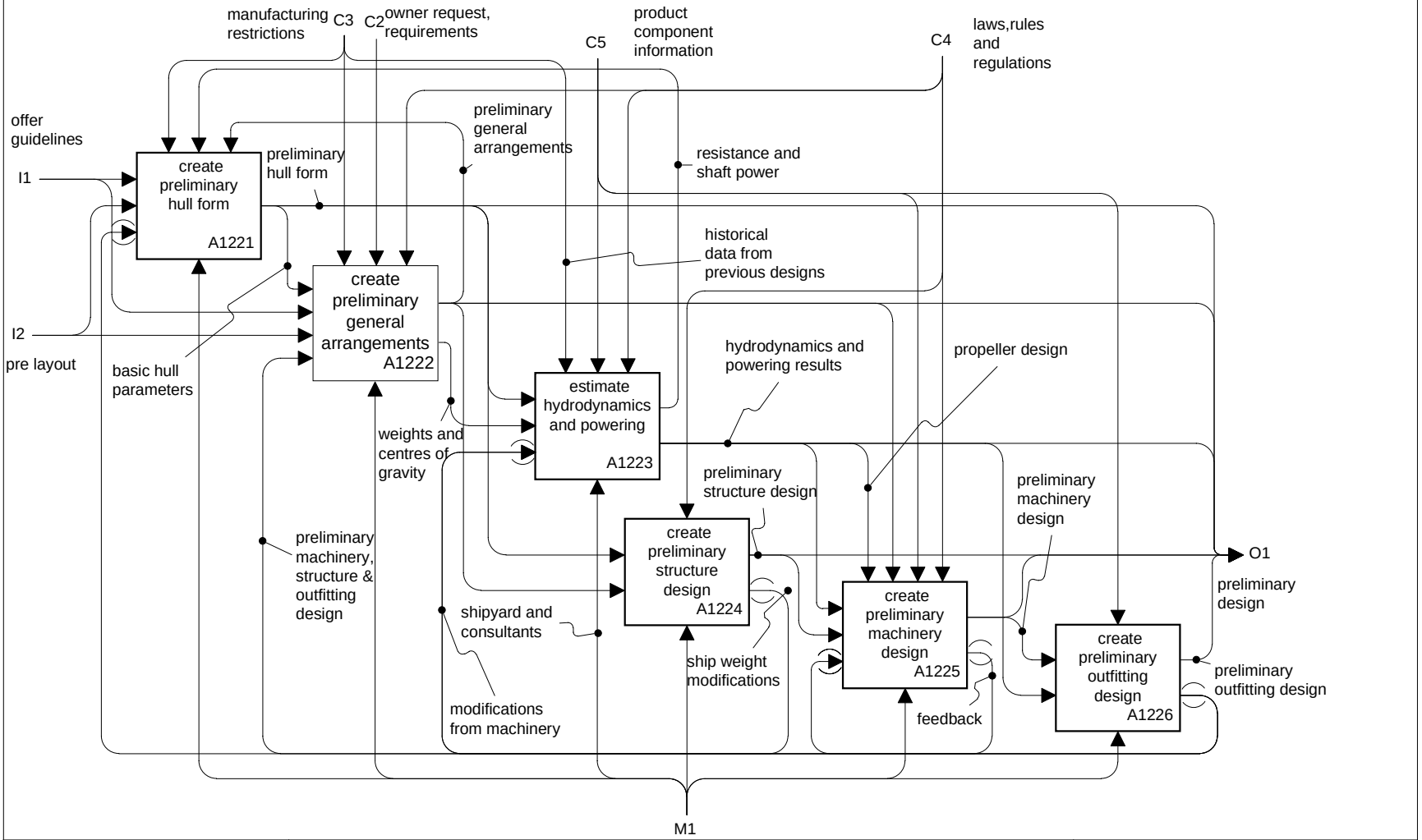
NODE: A1	TITLE: specify ship	NUMBER: P. 3

USED AT:	EDITOR: John Kendall	DATE: 22/05/96				CONTEXT:
	PROJECT: STEP Shipbuilding Common AAM	REV: 2.0	x	WORKING	READER DATE	
				DRAFT		
	NOTES: 1 2 3 4 5 6 7 8 9 10			RECOMMENDED		
				PUBLICATION		



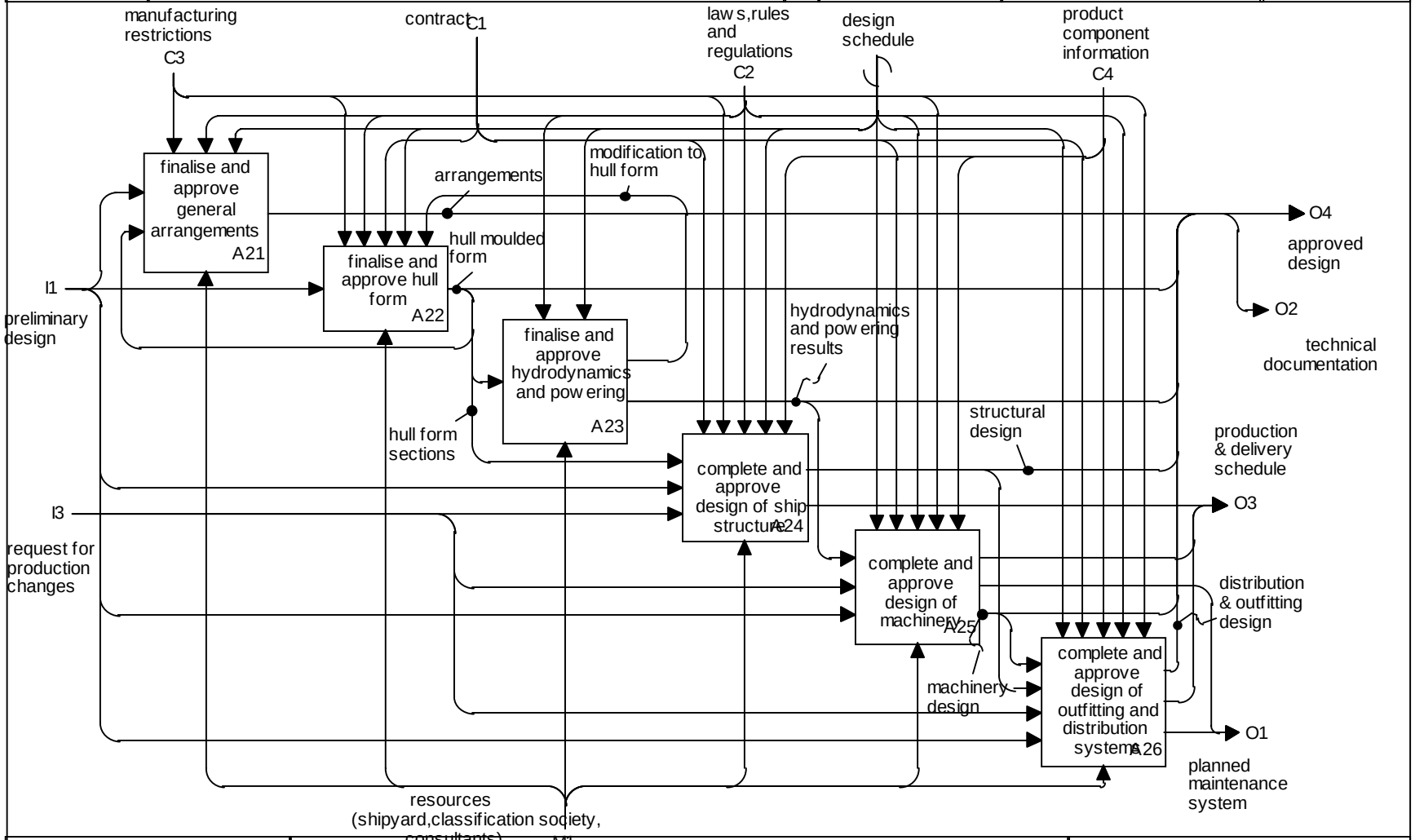
NODE: A12	TITLE: prepare bid	NUMBER: P. 4

USED AT:	EDITOR: John Kendall	DATE: 22/05/96	x	WORKING	READER DATE	CONTEXT:
	PROJECT: STEP Shipbuilding Common AAM	REV: 2.0		DRAFT		
	NOTES: 1 2 3 4 5 6 7 8 9 10			RECOMMENDED		
				PUBLICATION		



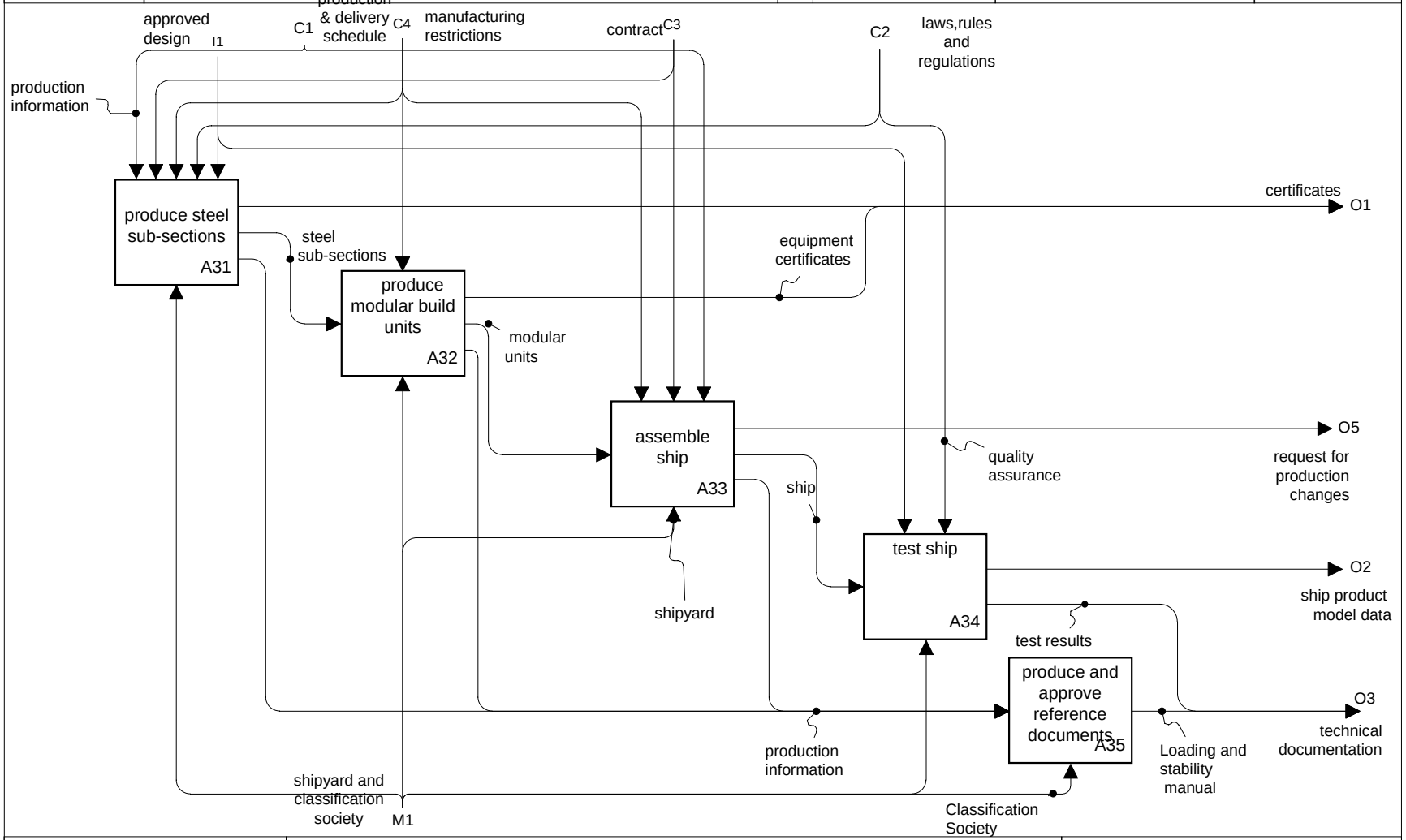
NODE: A122	TITLE: create preliminary design shipyard	NUMBER: P. 5

USED AT:	EDITOR: John Kendall	DATE: 22/05/96	x	WORKING	READER	DATE	CONTEXT:
	PROJECT: STEP Shipbuilding Common AAM	REV: 2.0		DRAFT			
				RECOMMENDED			
	NOTES: 1 2 3 4 5 6 7 8 9 10			PUBLICATION			



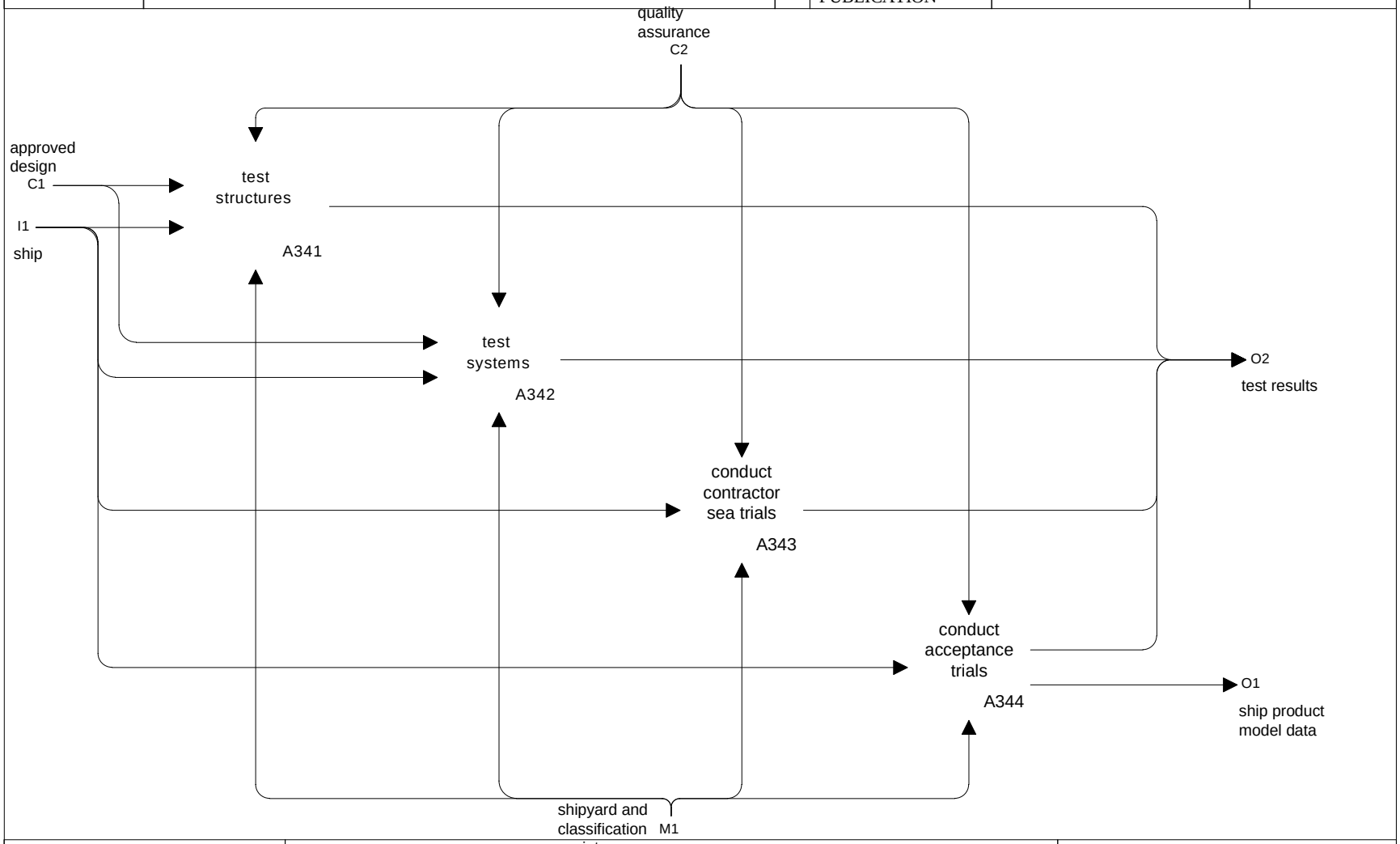
NODE: A2	TITLE: complete and approve ship design	NUMBER: P. 6

USED AT:	EDITOR: John Kendall	DATE: 22/05/96	x	WORKING	READER	DATE	CONTEXT:
	PROJECT: STEP Shipbuilding Common AAM	REV: 2.0		DRAFT			
	NOTES: 1 2 3 4 5 6 7 8 9 10			RECOMMENDED			
				PUBLICATION			



NODE: A3	TITLE: produce and inspect a ship	NUMBER: P. 7

USED AT:	EDITOR: John Kendall	DATE: 22/05/96				CONTEXT:
	PROJECT: STEP Shipbuilding Common AAM	REV: 2.0	x	WORKING	READER DATE	
				DRAFT		
	NOTES: 1 2 3 4 5 6 7 8 9 10			RECOMMENDED		
				PUBLICATION		



NODE: A34	TITLE: test ship	NUMBER: P. 8

5.5 EXISTING BUILDING BLOCKS FOR THE SHIPBUILDING APS

Building Block name	215	216	217	218	226	SCM V1.0
definitions	u	u	u	c	u	f - 1.21
generic_product_structures	u	u	u	c	u	f - 1.12
representation_resources	u	c	u	u	u	f - 1.10
connection_topologies			c			d - 1.1
product_structure_by_assembly				c	u	d - 1.12
product_structure_by_system	u			c		d - 1.8
approvals	u	u	u	c	u	u -1.7
changes	u	u	u	c	u	u - 1.3
designation_characteristics		c		u	u	u - 1.19
dimension_characteristics		c		u		u - 1.18
local_co_ordinate_systems_with_station_reference	u	u	u	c	u	u - 1.7
moulded_form_lines	u	c		u		u - 1.12
moulded_form_points	u	c		u		u - 1.14
moulded_form_representations	u	c		u		u - 1.16
moulded_form_surfaces	u	c		u		u - 1.12
ship_types	?	u	?	?		u - 1.2
features			u	c		u/d 1.9
parts				c		ud -1.4
date_time_resources		u		c	u	uf - 1.6
documents		u		c	u	uf - 1.10
events	u	u	u	c	u	uf - 1.4
external_references		u	u	c		uf - 1.12
geometry_model_resources	u	c	u	u	u	uf -1.7
geometry_resources	u	c	u	u	u	uf - 1.6
global_axis_characteristics	u	c	u	u	u	uf - 1.12
local_co_ordinate_systems	u	u	u	c	u	uf - 1.6
measures	u	u	u	c	u	uf - 1.12
organisation_resources		u		c		uf - 1.8
p41_resources	u	u	u	u	u	uf - 1.3
p42_resources	u	u	u	u	u	uf - 1.4
p43_resources	u	u	u	u	u	uf - 1.3
ships	u	u	u	c	u	uf -1.10
support_resources	u	u	u	c	u	uf -1.10
topology_resources	u	c	u	u	u	uf - 1.8
versions	u	u	u	c	u	uf - 1.11
materials			u	c	u	uu - 1.10
spacing_grids	u	c	u	u	u	uu - 1.15
advanced_boundary_representations			u	c		
approval_constraint_networks	u	u	u	c	u	
arrangement_descriptions	c					
arrangement_relationships	c					
assembly_manufacturing_definitions				c		
attachments			c			
cargo_assignments	c					

cargoes	c					
class_approvals_structure				c		
coatings	c					
compartment_design_definition	c					
compartment_properties	c					
compartments	c					
constructive_solid_geometry			c			
control			c			
corner_cutout_design_definitions				c		
corner_cutouts				c		
damage_stability						
design_loads				c		
distribution_representations			c			
edge_based_wireframes				c		
edge_cutout_design_definitions				c		
edge_cutouts				c		
engineering_parts			c			
equipments			c			
expansion_joints			c			
fasteners			c			
feature_design_definitions				c		
fittings			c			
freeboard_characteristics	c					
hull_cross_section_design_definitions				c		
hull_cross_sections				c		
hull_structural_survey				c		
hydrostatic_displacement_conditions		c				
hydrostatics		c				
instruments			c			
intact_stability		c				
interconnections			c			
interfaces			c			
interior_cutout_design_definitions				c		
interior_cutouts				c		
library_references						
lightship_weight	c			u		
loading_conditions	c					
machinery_survey					c	
maintenances			c	u	u	
miscellaneous_survey			c			
moulded_form_characteristics		c				
moulded_form_regions		c		u		
object_classification			c			
offset_table_representations		c				
operating_conditions			c			
organisation_numbers			c			
p501_aic						
p502_aic						
p507_aic	?	?	?	?	?	
p508_aic	?	?	?	?	?	

p509_aic	?	?	?	?	?	
p510_aic						
p511_aic	?	?	?	?	?	
p512_aic						
p514_aic						
p515_aic						
panel_design_definitions				c		
panels						
paths			c			
performances			c	u	u	
pipe_assemblies						
pipe_items			c			
pipe_run_state			c			
pipe_stress			c			
piping_parts			c			
piping_ports			c			
piping_specifications			c			
piping_system_tests			c			
piping_systems						
planned_maintenances			c	u	u	
product_structure_by_connectivity						
profile_endcuts				c		
repairs			c	u	u	
structural_connections_and_joints				c		
structural_edge_cutouts				c		
structural_features				c		
structural_openings				c		
structural_part_design_definitions				c		
structural_part_manufacturing_definitions				c		
structural_part_manufacturing_edge_features				c		
structural_part_manufacturing_layout_features				c		
structural_part_manufacturing_nc_features				c		
structural_part_manufacturing_production_templates				c		
structural_part_manufacturing_raw_material_stock				c		
structural_part_manufacturing_shape_representation				c		
structural_part_survey_definition				c		
structural_parts				c		
structural_system_design_definitions				c		
structural_systems				c		
survey_inspection				c	u	
survey_status				c	u	
tonnage	c					
valves			c			
weight_data	c			u		
weld_definitions				c		
welded_joints				c		
KEY:						

1 Date of the latest version of that Building Block; 2 The person responsible for maintaining the Building Block; 3 A Building Block is created by one AP, but may be re-used by others:
c = created by , u = used by; f = framework; d = domain model; uf = used by framework; ud = used by domain model; uu = used by utility